Objective ICT-2007.1.1

The Network of the Future

Project 216041

"4WARD – Architecture and Design for the Future Internet"

D-4.2
In-Network Management Concept

Date of preparation: **09-03-31**
Start date of Project: **08-01-01**
Project Coordinator: **Henrik Abramowicz**
Ericsson AB

Revision: **2.0**
Duration: **09-12-31**

Document Properties:

Document Number:	FP7-ICT-2007-1-216041-4WARD / D-4.2
Document Title:	**In-Network Management Concept**
Document responsible:	NEC
Author(s)/editor(s):	Editors: Giorgio Nunzi, Dominique Dudkowski (NEC) Authors: Martin Franzke and Gerhard Hasslinger (DT), Catalin Meirosu (EAB), Ilka Miloucheva, Michael Kleis, Fabian Wolff, Rudolf Roth, Thomas Hirsch (Fraunhofer), Susana Sargento (IT), Rolf Stadler, Mads Dam, Fetahi Wuhib, Dan Jurca, Alberto Gonzalez (KTH), Chiara Mingardi (NEC), Frank-Uwe Andersen (NSND), Vitor Mirones (PTIN), Daniel Gillblad, Rebecca Steinert (SICS), Attila Katona (SROM), Avi Miron (Technion) , Sławomir Kukliński (TPSA), Christopher Foley, Eamonn Power (TSSG), Virgil Dobrota, Andrei Bogdan Rus, Emanuel Puschita (TUCN)
Target Dissemination Level:	PU
Status of the Document:	Final
Version	2.0

This document has been produced in the context of the 4WARD Project. The research leading to these results has received funding from the European Community's Seventh Framework Programme ([FP7/2007-2013] [FP7/2007-2011]) under grant agreement n° 216041

Abstract:

This deliverable reports on our progress towards developing the paradigm of In-network Management (INM), a clean-slate approach to network management, aimed specifically at the effective management of large, dynamic networks, where a low rate of interaction between an outside management entity and the network will be required. The idea is that management tasks are delegated from management stations outside the network to a self-organizing management plane inside the managed system. This is enabled through decentralization, self-organization, embedding of functionality and autonomy. Under this paradigm, the managed system executes functions –locally or end-to-end– on its own and performs, for instance, reconfiguration or self-healing in an autonomic manner. It reports results of its action to an outside management system or triggers exceptions if intervention from outside is needed.

The deliverable describes the main results of WP4 achieved during the first year of the project and it complements deliverable D-4.1, which presents use cases illustrating the potential of INM capabilities. It contains a first version of the INM framework design, which defines the structure of the management plane inside the network, supports the embedding of management functions and provides reusable components to compose collaborating self-managed entities. Second, it presents a set of algorithms and concepts developed for real-time management, with emphasis on distributed monitoring in large-scale dynamic environments. Third, it reports on work that demonstrates the feasibility of rapid re-configurability for selected management algorithms and functions under the INM paradigm. Extensive simulations have demonstrated the effectiveness of the approach and have quantified key trade-offs.

The report ends with an outlook of the project plans for the second year, where effort will be devoted to proving the technical feasibility through prototype implementation of selected functions and integration of the work with that other WPs.

Keywords:

Future Internet, in-network management, scalable and robust management systems, self-management, self-organization, clean slate design, architectural principles and elements, real-time monitoring, topology learning, situation awareness, anomaly detection, network management benchmarking, controlling performance trade-offs, self-adaptation, autonomic management, event distribution, information storage and retrieval capabilities

Table of Contents

Executive Summary

This deliverable reports on our progress towards developing the paradigm of **In-network Management (INM)**, a clean-slate approach to Internet management, aimed specifically at the effective management of large, dynamic networks, where a low rate of interaction between an outside management entity and the network will be required. The idea is that management tasks are delegated from management stations outside the network to a self-organizing management plane inside the managed system. The INM approach thus involves embedding management intelligence in the network, enabled through decentralization, self-organization, embedding of functionality and autonomy. Under this paradigm, the managed system executes functions –local or end-to-end– on its own and performs, for instance, reconfiguration or self-healing in an autonomic manner. It reports results of its action to an outside management system or triggers exceptions if intervention from outside is needed.

The deliverable describes the main results of WP4 achieved during the first year of the project and it complements deliverable 4.1, which presents use cases illustrating the potential of INM capabilities. It contains a first version of the INM framework design, which defines the structure of the management plane inside the network, supports the embedding of management functions and provides reusable components to compose collaborating self-managed entities. Second, it presents a set of algorithms and concepts developed for real-time management, with emphasis on distributed monitoring in large-scale dynamic environments. Third, it reports on work that demonstrates the feasibility of rapid re-configurability for selected management algorithms and functions under the INM paradigm. Extensive simulations have demonstrated the effectiveness of the approach and have quantified key trade-offs.

The report ends with an outlook of the project plans for the second year, where effort will be devoted to proving the technical feasibility through prototype implementation of selected functions and integration of the work with that other WPs.

Terminology

Architectural elements of INM: collective term for the following five architectural elements of the INM framework: management capabilities, functional components, INM entities, INM protocol, and INM registry.

Atomic service: an atomic service is a service whose implementation is self-contained and does not invoke any other services.

Capability: the potential of one or more resources (logical or physical) to be able to achieve specific effects or actions, or declared goals and objectives.

Capability Level Statement (CLS): a declaration by one or more resources stating the capability of the resource(s).

Composite service: a composite service is a service whose implementation calls other services to provide and fulfil its obligations in terms of service provisioning. In other terms, the services that are created by the federation of other services are called composite services.

Contract: a contract is an agreement between two parties, generally positing that one party performs a service or provides a product in exchange for compensation from the other party. The contract additionally defines consequences in case the agreement is not fulfilled, and if there are any factors that may justify not fulfilling parts of the agreement.

Dedicated management entity (ME): an INM entity that bundles management capabilities that are not tied to any self-managing entity.

Dedicated management functional component (dmFC): a functional component that encapsulates only management functionality (example: cross-layer neighbour table).

Entity: a network element that can be viewed as a network node. There are different types of entities according to their capabilities (e.g. access entity, core entity, distribution entity).

External management capability (exMC): any management capability that is located external to any INM entity or functional component.

Functional component (FC): designation for any implementation-level component that encapsulates a network service and/or management functionality within a single entity.

Governance: the process of refining an end-user service description (entered by the business service manager (BSM)) to Service Level Agreements (SLAs), which provide the objectives for composite and atomic services delivered by self-managing entities.

Governance domain: the domain (network elements, managed functions, subordinate SEs) over which an entity (for example an SE, the BSM, or an organization) has jurisdiction.

I-NAME: (In-Network Autonomic Management Environment): autonomic reacting environment that generates a set of management messages for resource control and maintenance accordingly to the dynamically changing network's configuration.

Inherent management capability (ihMC): management capability that may be only present within self-managing INM entities / functional components.

INM artefacts: collective term for the architectural elements, technical framework, and algorithms.

INM entity: type of architectural element comprising dedicated management entities (MEs) and self-managing entities (SEs).

INM principles: collective term for the five extremal clean slate principles that underlie INM, comprising the intrinsic, inherent, autonomous, abstraction, and transition principle.

INM protocol: the protocol specific to in-network management that mediates management between INM entities.

INM registry: architectural element that contains the necessary functionality to translate between INM entities and functional components.

INM transitional degrees: design space comprising the dimensions of the degree of embedding, autonomy, and abstraction that lead towards the extremal INM paradigm.

Integrated management capability (igMC): management capability that may be present in both self-managing and dedicated INM entities / functional components.

Lifecycle of self-managing entities: defines the set of actions performed during the introduction of a self-managing entity into the self-organizing management plane, including 1) design and implementation, 2) deployment and activation, 3) operation and reconfiguration, and 4) termination and deregistration of a self-managing entity.

Management capability (MC): an abstraction for fine-granular management capabilities, the building blocks for composing any more complex management functions.

Management interfaces: collective term for the organization and collaboration interface that are supported by management capabilities, functional components and INM entities.

Management plane (MP): an abstraction of management functionality in a network or system. In the case of INM, the management plane resides inside the self-managing system, which means it is realized as part of the network infrastructure.

Node: a node is a physical entity that contains a set of management capabilities, functional components, and an INM registry.

Provider: entity who is responsible for and operates the network infrastructure (physical or virtual) or services in general.

Report generator: in the context of event handling, it is the role assumed by an INM entity or functional component which is able to generate a report.

Report handler: in the context of event handling, it is the role assumed by an INM entity or functional component which is able to handle a report.

Self-management: the process by which a network/system (or part of it) governs its own behaviour and operation without human intervention.

Self-management by objective: the process by which a network/system (or part of it) governs its own behaviour and operation without human intervention, given a set of objectives that it gets through automatic refinement of a formalised end-user service description.

Self-managing entity (SE): a self-managing entity is a component of a system that is self-managed by objective and can autonomously perform a series of management-related tasks, such as self-configuration, self-healing, automatic discovery of peer entities and negotiation of service level agreements.

Self-managing functional component (smFC): a functional component that encapsulates both, a network service and management functionality (examples: Generic Path, NetInf).

Separated management capability (spMC): designation for integrated management capabilities in a dedicated management functional component, which, from the perspective of another self-managing functional component (smFC), appear separated from that smFC.

Service: a mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistently with constraints and policies as specified by the service description.

Service description / service specification: a description of a service in terms of interfaces, operations, semantics, dynamic behaviours, policies, and qualities of service, including the minimum performance level.

Service Level Agreement (SLA): a contractual specification provided to a consumer of a service describing the service and the quality associated to the service.

Structure: a self-organized set of INM entities sharing the same requirements.

1 Introduction

Work Package 4 (WP4) is working towards the definition of novel management instruments to operate the future Internet. Following a clean slate approach, it defines a new architecture to support reliable and scalable management operations as well as the algorithms to implement them as distributed real-time functions.

1.1 Motivation of Document

The work dedicated towards network management in 4WARD is triggered by limitations becoming apparent in current network management. As documented in WP4's previous deliverable D4.1 [1], today's centralized network management will no longer be applicable to the large-scale networks and services foreseen in the future; therefore new approaches for management architectures are needed.

4WARD In-Network Management (INM) addresses these challenges by introducing a thin, pervasive layer which performs core management functionalities already inside the network, but which can be complemented by additional management functionality outside the network where necessary.

This document is a deliverable summarizing the main concepts developed by WP4 during the first year and follows the previous deliverable, describing scenarios where traditional network management fall short. The concepts elaborated in this document will be further elaborated in final design document, and then evaluated and implemented through a prototype.

1.2 General Presentation of the 4WARD Project

4WARD aims to increase the competitiveness of the European networking industry and to improve the quality of life for European citizens by creating a new generation of dependable and interoperable networks providing direct and ubiquitous access to information, evolving and replacing today's Internet paradigms. This will pave the ground for more advanced and more affordable communication services for the next decades, beyond 2015.

Unlike most other EU projects in this area, 4WARD is committed to a 'clean slate approach' to address these issues.

The term 'clean slate approach' stands for a coherent solution that breaks the current network's stagnation imposed by the need to 'support current technologies and solutions'; It is the answer to the following question: 'With what we know today, if we were to start again from scratch, how would we design a global communications infrastructure?'

Hence, the overall objective of 4WARD is to create a framework of innovative networking models that together will define the direction towards a 'Network of the Future'. In the current (first) phase, focus of the project is on exploring specific technology directions with identified key potential for the future and assessing the feasibility of concepts in first validation studies.

The **key technology research areas** identified for the 4WARD approach are:

- **Architecture framework**: 4WARD will improve the ability to design inter-operable and complementary families of network architectures and develop an integrated framework to represent, design, implement and operate network architectures that all belong to a common family of interoperable network instances.
- **Virtualisation**: 4WARD will enable the co-existence of multiple networks on common platforms through carrier-grade virtualisation of networking resources. 4WARD will provide means to support the instantiation and dependable inter-operation of different networks on a single infrastructure in a commercial setting.
- **In-network management**: 4WARD will enhance the utility of networks by making them self-managing. 4WARD aims for an embedded 'default-on' management capability which is an inseparable part of the network itself. This capability will generate extra

value in terms of guaranteed performance in a cost effective way, and will enable the networks to adjust themselves to different sizes, configurations and external conditions.

- **Generic connectivity**: 4WARD will supersede the current point-to-point forwarding and routing scheme by a new paradigm of functionally rich communication paths, increasing the capacity and dependability of networks composed of mixed technologies (wired and wireless) and supporting applications that require more than today's point-to-point dissemination pattern. These new communication paths will deal with mobility, security and Quality of Service requirements in an integrated way.
- **Content-centric network of information**: Finally 4WARD will revolutionise application support by a new information-centric paradigm in place of the old host-centric approach: 4WARD will create a new *network of information* paradigm where information objects have their own identity and are no longer bound to specific hosts.

These 4WARD technology solutions shall embrace the full range of current and future network technologies. 4WARD technical results will be disseminated in the context of non-technical drivers to bridge the gap between innovative research results and utilization for the benefit of the economy and the society at large.

1.3 Presentation of Related Work Packages

In this section we present key relations of WP4 with other WPs within the 4WARD project and we briefly state for each work package the relation to WP4.

WP1: Business Innovation, Regulation, and Dissemination – BIRD

WP1 follows the idea that research on technology should be accompanied by research on the context of its intended usage. The WP covers non-technical topics and focuses especially on society, business and governance issues. WP4 provides technical means to support novel business models in the Future Internet and to enforce regulatory changes.

WP2: New Architecture Principles and Concepts – NewAPC

WP2 is exploring the development of a design process for combining existing, or specifying and generating new networks with customized architectures. WP4 is developing the following functions that can be mapped into WP2 framework:

- A set of management operations.
- Registration and access mechanisms for embedded self-descriptive management functions.
- Scheme, strategies, and protocols for collaborative monitoring, self-optimizing, and self-healing.

WP3: Network Virtualization – Vnet

WP3 will investigate the approach of using virtualisation to enable flexible and innovative networking architectures. Virtualization allows an evolution of communication technology while largely reusing deployed infrastructure. It further provides a general framework for network sharing: providing different networking services of different network service providers on a common physical infrastructure. WP4 is developing the following functions for Vnet management:

- Monitoring
- Adaptation algorithms, which most likely will run over a single Vnet
- Bootstrapping methods will be delayed to the beginning of the joint Task TC34.

WP5: Forwarding and Multiplexing for Generic Paths – ForMux

WP5 is focused on development of a model and underlying mechanisms for Generic Path abstraction that would provide unified means for applications to address transport capabilities of the network. WP4 provides the following functions:

- Methods for optimization across different elements
- Support of business models.

INM has the responsibility of network-wide management. As such, load balancing is a key function, which must be addressed in a top-down manner. As congestion is detected or is proactively predicted to be developed, some traffic must be routed away from this area. Since route setup is a function mainly performed by a GP, INM can better support routing functions to accommodate load-balancing and other specific management objectives, and enforce them into the WP5-routing entity through corrective actions.

WP6: Network of Information – NetInf

The overall objective for WP6 is to investigate a new information architecture, called NetInf, where information retrieval and storage act on the objects themselves rather than on the nodes. The joint Task TC46 produced a milestone describing the relation between the two WPs in detail. This is discussed in detail in Section 6.2 and 6.3. It is assumed that management of the NetInf objects themselves is inherently done by the objects (e.g. instantiation of the object). WP4 addresses the network management aspects that go beyond the management of the NetInf object management capabilities (e.g., managing a service, topology discovery, the traditional FCAPS capabilities).

1.4 Objective of Document

This document contains the main concepts for the design of the new INM framework. It identifies the specific requirements for designing the management capabilities, based on which a framework for in-network management will be presented. Aspects of self-organization of the distributed functions, the collaborative management support, as well as the registration of management functions are described as well. The key algorithmic studies for real-time and adaptation mechanism in INM are presented and first results discussed.

These concepts are the intermediary step towards the definition of the complete INM design and its subsequent evaluation and implementation. This will be achieved in the second year of the 4WARD project.

1.5 Structure of Document

After the introduction given in this section, the document is structured as follows:

Chapter 2 summarizes the motivation for INM and explains the business values of INM through self-management.

Chapter 3 introduces the main concepts of the INM framework. The main design principles for the clean slate approach are identified. Based on that, the main components are introduced and self-organization mechanisms are presented. Finally the chapter concludes with a couple of examples on instantiating algorithms in the framework.

Chapter 4 introduces algorithms for real-time monitoring and situation awareness. First different schemes for the generation of global aggregates are presented, then the mechanisms for situation awareness and anomaly detection.

Chapter 5 discusses adaptation schemes of INM. First, an analysis on the different trade-off about distributed and centralized approaches are presented. Then, a set of adaptation schemes for the INM management plane as well as network resources in 4WARD is presented.

2 Motivation for In-Network Management and Scope of Work

2.1 Limitations of Traditional Approaches

With current technologies, management functions typically reside outside the network in management stations and servers, which interact with network elements and devices via management protocols, in order to execute management tasks, including fault, configuration, accounting, performance, and security management. Most of these tasks are performed on a per-device basis. During network operation, for instance, a management station periodically polls individual devices in its domain for the values of local variables, such as devices counters or performance parameters. These variables are then processed on the management station to compute an estimate of a network-wide state, which is analyzed and acted upon by management applications. This paradigm of interaction between the management system and managed system underlies traditional management frameworks and protocols, including SNMP, TMN and OSI-SM [23].

Over the past 20 years, this paradigm has proved fairly successful, specifically for networks of moderate size, whose configuration rarely change and whose states evolve slowly and thus do not require rapid intervention by an outside system. These assumptions, however, do not hold for many of today's emerging networks. We envision that the future Internet, particularly with its wireless and mobile extensions, will be a system whose management domains can potentially include millions of network elements, whose configuration will change on a continuous basis, and whose state will be highly dynamic and thus must be available at control points with short delay.

Within the 4WARD project, we are primarily addressing those aspects of the traditional management paradigm that lead to poor scaling, to inherently slow reactions to changing network conditions and to the need for intensive human supervision and frequent intervention.

2.2 A Clean Slate Approach to Network Management

Following the traditional network management paradigm outlined above, two characteristics are prevalent in today's deployed management solutions:

- Network elements are "dumb" from a management standpoint. They offer simple, standardized interfaces for monitoring purposes. Interfaces for configuration and control are generally low-level and vendor-specific. In addition, network elements have little autonomy in learning about their environment and in making management decisions.
- Network elements interact directly with management entities outside the network. There is no interaction among network elements for management purposes. Consequently, management tasks, such as configuration or fault management are typically performed on a per-device bases.

Historically, these two assumptions were made, in order to have network elements of low complexity, to achieve a clear separation between the managed system that provides a service and the management that performs configuration and supervision, as well as to allow for simple, hierarchical structuring of management systems.

In the context of 4WARD, we mean by a **clean slate approach to network management** a way of envisioning and engineering management concepts and capabilities that do not rely upon the above assumptions but rather abandon them.

The approach we pursue within the 4WARD project is called **In-Network Management (INM).** Its basic enabling concepts are decentralization, self-organization, embedding of functionality and autonomy. The idea is that management tasks or subtasks can be delegated from management stations outside the network to a self-organizing management plane inside the network. The INM approach therefore involves embedding management intelligence in the network, or, in other words, making the network more intelligent. The managed system can

now execute certain functions--local or end-to-end--on its own. It can perform, for instance, reconfiguration or self-healing in an autonomic manner. It reports results of its action to an outside management system, or it triggers exceptions if intervention from outside is needed.

A central concept of INM is the above mentioned **management plane inside the network**. Such a plane serves as infrastructure for various kinds of management functions. It is made up of autonomous management entities that collectively provide management functions. It is self-organizing and adapts to changes in network topology, to failures and to external events. In order to realize the management plane, a management entity with processing and communication functions is associated with each network element or device, which, in addition to monitoring and configuring local parameters, communicates with peer entities in its proximity. The collection of these entities creates a thin layer of management functionality inside the network that can perform monitoring and control tasks end-to-end.

Recall that that the management plane the does not replace an outside management entity but rather complements such an entity. In any future network, there will be a need for setting business objectives and policies from outside, and for possible intervention, if things go wrong. The goal of INM is to significantly reduce the rate of interaction between an outside management entity and the network, not to make the network completely autonomous—a proposition we find neither feasible nor desirable.

The need for distributing management tasks has been recognized before and has been studied in the research community since the mid 1990s. Concepts like management by delegation, mobile agents, and distributed objects have been developed with the goal of making network management systems more efficient, better scalable and less complex (cf. [36]). Within the same time frame, new engineering concepts in networking and telecommunications, namely active networking and programmable networks have appeared, aimed at simplifying the introduction of new functionality into networks ([21],[22]).

The novelty of the INM approach is that it combines and refines, in a specific way, some of the above concepts and moulds them into a new network management paradigm that centers around embedding of management functionality into the network elements, decentralization of management tasks, self-organization of the management infrastructure, and autonomy of management entities.

The potential benefits of this paradigm that we are developing and evaluating in 4WARD include the following properties:

- A high level of scalability of management systems, for instance, in terms of small execution times and low traffic overhead in large-scale systems. This will allow for effective management of large networks.
- Fast reaction in response to local and external events. This will increase the adaptability of the network to various kinds of faults, configuration changes, load changes, etc.
  Together with embedded functions, this will lead to with a high level of robustness of the managed system.
- A high business value for INM technologies through reducing capital and operational expenditures (cf. Section 2.4 below).

A possible drawback of this paradigm is that processing resources for management purposes must be available in the network elements (or in the managed system), which will potentially increase cost and network complexity. The challenge is to demonstrate that the INM approach*, on balance*, can provide significant benefits over current technology.

2.3 The Scope of Network Management in 4WARD

It is often useful to understand the scope of network management by using the functional model defined by ITU-T and ISO, which partitions network management into five functional areas (referred to by the acronym FCAPS): Fault and Configuration Management, Accounting

Management & User Administration, Performance and Security Management [24]. A somewhat broader definition of network management can be found in [25], which says that "Network Management refers to the activities, methods, procedures, and tools that pertain to the operation, administration, maintenance and provisioning of networked systems." It is clear that 4WARD cannot provide a comprehensive treatment of all these aspects, but must focus on key components that are relevant to developing and evaluating the INM paradigm.

From the five functional areas, we have, so far, concentrated on aspects of performance and fault management (Chapter 4 and 5) and plan to extend our activities into specific issues in configuration and security management. Regarding the definition in [25], we have given weight to technical contributions in support of operation (Chapter 4 and 5), while, at the same time, developing the INM framework (Chapter 3) with the vision that it can ultimately support all aspects mentioned in [25].

An area we have given attention to is real-time monitoring or, more general, situation awareness, for several reasons. First, monitoring, i.e., the process of acquiring state information from a network or networked system, is fundamental to system operation regarding all five functional areas. Second, monitoring functionality is generic (in contrast to, say, configuration), in the sense that it can directly support the management of various technologies, specifically those being developed within 4WARD, i.e., network virtualization, generic paths and network of information.

2.4 The Business Value of In-Network-Management

INM addresses state-of-the-art design principles and processes for the management of the future Internet. They facilitate to launch new services and in addition help to reduce the effort and costs of service and network provisioning. Since a quantitative prediction of the INM business value for developing new services seems speculative, we focus on describing main cost saving effects through INM for operational (OpEx) and capital (CapEx) expenditure. Driving forces of the expected savings and business support can be seen in:

- Self-organization and automated processing, reducing the need for manual intervention in current network operation;
- Distributed monitoring and control for better situation awareness, resulting in faster and more precise proactive and repair processes;
- Virtualization concepts which include management in a common framework;
- Improved control and reporting functions for business management.

Savings in CapEx are expected through combined situation awareness and optimization allowing for more efficient network resource usage. Since current preferences for over-provisioned networks also contribute to wastage of energy in underutilized electronic equipment, optimized resource management supports CapEx saving together with a trend to green ICT.

We give first a brief overview of the relevant categories in business value.

2.4.1 Relevant Categories in OpEx and CapEx

Operational and capital expenditures are interconnected issues [34]. It depends on the network environment whether OpEx or CapEx represents the major portion of the overall costs of the network and service provisioning [27][29]. Automated technologies and functions to facilitate management tasks may initially shift costs from OpEx to CapEx but in the long term will reduce the total costs as simplifying standard solutions. The level of achievable self-organization is different in fixed backbone and access network types, in wireless, mobile, sensor networks or in heterogeneous networks.

Network layers are another criterion for differentiation. Virtualization concepts, which are investigated in WP3, can provide a completely self-organizing environment for some task on top of physical network layers. Developments towards virtual concepts are already used in

application layer overlays, e.g. peer-to-peer networking, whose economic aspects are a main topic of the EU project SmoothIT, see Annex C.

INM is focused on processes in operational networks. Therefore first installation costs in OpEx as well as CapEx for setting up a network, buying equipment etc. are not considered. The costs occurring in an operational network can be divided into the following categories related to main management tasks and processes [32]:

Continuation of normal network operation: The cost to keep the network running in a failure free situation including space, power supply, leasing equipment, e.g. fiber rental etc.

Maintenance and monitoring: The cost to maintain the network and to operate the network with awareness of failure events based on monitoring of the network and its services

Failure handling and recovery: Failures in the network have to be repaired on a case by case basis triggered by alarms, if this cannot happen in routine operation. Reparation may lead to actual service interrupts depending on the protection scheme.

Planning and updating of an operational network: This category includes all planning performed in an existing network which is up and running, resulting in long term upgrade processes for increasing traffic and resource demands as well as short term workarounds and (re-)optimization, upgrades or replacement of outdated software and hardware components.

Service management and provisioning: Processes set up to provide and control previously negotiated services to customers, usually defined via service level agreements (SLA)

Business management, marketing, pricing, sales: Business and service managers govern the network to support a service portfolio through a set of business decision processes. They use business level policies at the top level when developing new end-user services.

INM has direct influence on the first four categories and interacts with service and business management through predefined interfaces and processes. Studies on the distribution of OpEx for network operators [29][31] attribute roughly 27% to marketing and sales, 24% to customer and IT support services, 22% to network elements and 27% to interconnection and roaming. Energy costs of the network equipment should be included in the network element part. Personnel or corresponding room rental costs also should include energy consumption. The portion for interconnection and roaming is higher for mobile and lower for fixed operators. Interconnection costs also differ with the size of ISPs. Large tier-1 ISPs often provide Internet connectivity for smaller ISPs but can profit from peering contracts with them.

2.4.2 OpEx reduction through self-managing INM instances

In nowadays heterogeneous networks, different domains establish their degree of embedding, autonomy and abstraction, depending on the technology, applications and administrative goals. Wherever a higher autonomy level or more flexible adaptation is achieved, expenses for manual intervention are reduced with consequences for OpEx [26][30].

Centralized management in large scale networks is often subject to uncontrolled floods of alarms being forwarded to the NOC in failure cases. Self-managing entities in INM are expected to enforce efficient local control loops which can filter and forward more consolidated reports instead of spontaneous data records. INM can aggregate control messages within entire sites and areas. It will not be necessary to have every network element connected to network management systems. While this can reduce overhead and facilitate failure analysis in large scale networks, it is a precondition for deploying small home or building networks, which have to work in an autonomic, OAM-free manner and otherwise have to generate failure reports to be understood by untrained people.

2.4.3 CapEx reduction through situation awareness and fast adaptability

Giving preference to a higher autonomy level, INM can also profit from ICT technology trends towards steadily improving performance. As a side effect of increasing bandwidth, monitoring and control cycles can be executed at higher frequency, resulting in immediate and more precise situation awareness for distributed network functions. Consequently, resource

optimization, load balancing and rerouting can adapt to traffic shifts and prospective failure scenarios in shorter response cycles, which makes them more reliable and robust.

As an example, we consider traffic engineering on broadband access platforms, which has to cope with link upgrading processes for increasing Internet traffic and has to take care of failure situations by providing backup paths as discussed in Section 5.1.1. (G-)MPLS or Ethernet with multiple spanning trees are technologies that allow to establish load-balancing traffic paths. They can be pre-computed for modified topologies including relevant failure scenarios, but it is still time consuming to reconfigure the paths to circumvent a failed link in an operational network. Therefore additional capacity is usually provisioned to overcome non-optimized intermediate stages after topology changes due to failures or upgrades.

INM functionalities for faster situation awareness and adaptability through automated online processes can reduce those time gaps subject to unbalanced or instable load in the network and, as a consequence, allow to reduce over-provisioning for such situations. Experience with traffic engineering in IP backbones [28] has shown that a 20%-30% increase in utilization – or a corresponding decrease in provisioned bandwidth of IP routers and transmission equipment – is possible when the network can be kept in optimized load-balanced regime by fast redirection of transport paths from overloaded resources.

2.4.4 INM support for business and service management

Business and service management relies on network management, e.g. for evaluation of the network status and performance over time. INM monitoring and automated reporting again improve the timeliness and precision of evaluated data and support SLA fulfillment etc.

As a long-term vision, self-adaptation and autonomous management may vertically integrate network resource monitoring, planning, administrative policies with business and market strategies. For example, pricing for access and IP services may be made dependent on temporary bottlenecks and restricted scalability of network resources. In the opposite way, marketing campaigns and new pricing for web services may automatically trigger support actions in resource provisioning and management for prospective shifts in demand.

2.4.5 INM in dynamic networks opening new applications and business areas

Dynamic network environments represent a challenging area where classical management and routing concepts fail, making a clean slate approach indispensable. With increasing dynamics, routing tables or centralized network state information for management are becoming more and more imprecise and invalid. MANETs, sensor networks and P2P overlays represent three different types of dynamic networks. The IETF recently has set up a working group on ROuting over Low power and Lossy networks (ROLL) to study how far existing routing protocols can be adapted and where new methods have to be set up for networks with highly unreliable nodes and links.

In addition to routing, network management has to be considered in a next step [35]. Therefore concepts of inherent network management are relevant to achieve fast local situation awareness and control, which can be evaluated and aggregated bottom-up into larger domains via distributed and gossip-based algorithms or reinforced learning as studied in Section 4.1. The scope of dynamic networks includes heterogeneous environments e.g. for home, building and urban networks or communication for recovery from catastrophes and military applications. When new concepts can be realized to manage and operate networks with a high level of dynamics, they should be flexible to also handle less dynamic scenarios and to support a centralized view of the network as far as possible.

INM may contribute in a long term perspective to a generalized network management framework spanning a wide range from distributed architectures for dynamic systems to centralized schemes for stable topologies[32][33]. In this way, migration steps towards self-organizing and autonomous components can be introduced bottom-up even in systems under central control.

3 Framework for In-Network Management

This chapter describes the framework for in-network management (INM) (short: INM framework) that supports management operations in the future Internet by the means of a highly distributed architecture. The main objective is the design of management functions that are located close to the management services, in most of the cases co-located on the same nodes; as target approach, they would be co-designed with the services. In this objective we identify the INM paradigm of *embedding management capabilities in the network*. The benefit of the resulting distributed architecture is the inherent support for self-management features, which promise to bring the business values discussed in Section 2.4.

The INM framework is the first step to achieve the aforementioned objective. In line with a clean slate approach, the framework proposes the fundamental principles and constructs that state how to design and operate concrete networks according to the INM paradigm. This is the basis for defining the algorithms in Chapter 4 and 5 as functions distributed in networks and, from them, to construct management operations through self-organizing mechanisms.

This chapter is structured as follows. Section 3.1 to 3.4 introduce the fundamental principles and elements of the INM framework that are the basis for designing and implementing embedded management processes. Section 3.5 and 3.6 describe mechanisms for the operation of embedded management processes with respect to a self-organizing management plane, including the discovery and structuring of management capabilities and information handling mechanisms. Section 3.7 concludes this chapter with a presentation of several examples that show how the INM framework is applied in concrete network scenarios.

3.1 High-Level Architecture for In-Network Management

The purpose of INM is to introduce cost-effective instruments to operate the future Internet. Before introducing the main concepts for the modular and distributed architecture of INM, it is necessary to understand who the main actors are and how they relate to the overall picture of 4WARD, in particular to the business framework and business models investigated by WP1 and WP3, respectively. For simplicity, we refer to a model comprising two roles, which will help to identify the main beneficiaries of INM, like illustrated in Figure 6-1:

- **Provider**: entity who is responsible for and operates network resources (either physical or virtual resources). The provider maps to any of the business roles discussed in D-1.1 [2], like infrastructure and service provider as well as virtual network operator. Each will require INM to operate its respective resource domains. To the provider, INM offers a set of interfaces to operate those resources and to respect SLAs with users.

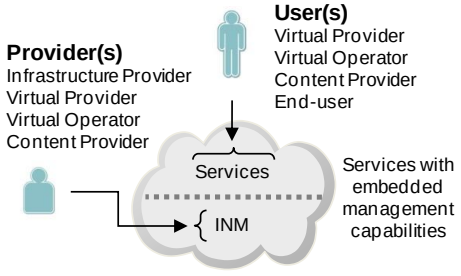


Figure 3-1: Actors in INM with respect to the business scenarios in D1.1

- **User**: entity who requests and uses providers' resources as services. The user is regarded in a general sense and can be assumed as end-user as well as provider, acting as customer in the value chains considered in the scenarios of D-1.1 [2].

Given the legal relationships between providers and users, an SLA is used to define the type of service delivered to the user and the guaranteed quality in delivering it. An SLA's technical description is therefore used as input by INM to configure services. Such descriptions can take the form of XML documents with well-defined negotiated network performances.

Since INM features management functions that are distributed to and embedded within the network elements, it is necessary to redesign management operations. The traditional FCAPS

model that builds on mostly separated vertical processes cannot easily be adopted to trigger embedded management capabilities. A new set of high-level interfaces must be introduced.

We have considered the following requirements that such a set of interfaces must satisfy: i) need for high-level abstractions to define management commands over large-scale networks, ii) support for objectives of self-configuration algorithms, iii) feasibility to map to local management capabilities, and iv) feedback mechanism for network redeployment.

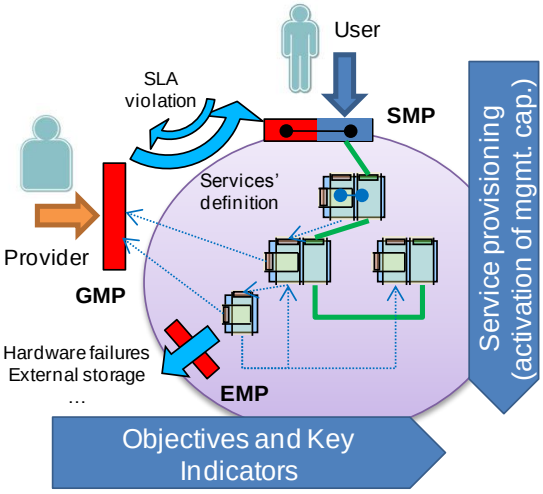


Figure 3-2: Management abstractions in INM

Based on these requirements, we have identified three types of management points (shown in Figure 3.2) that define the set of operations accessible to INM and the actors by which they are accessed. A single management point does not necessarily imply that it is implemented within a single network node. A high degree of distribution can still be adopted when implementing the interfaces to model the operations defined by that management point. The set of management points to access INM are the following:

- **Global management point (GMP):** It provides operations to define network-wide objectives and to construct global indicators from the network. A network operator uses the GMP's interfaces to instantiate services and to supervise the network. Differently from traditional management, GMPs provide objectives that are more abstract, less complex, and closer to business goals, and they map to distributed INM functions. Such abstractions are supported in the architecture through the INM entities and mapped to the organizational interface.
- **Service Management Point (SMP):** It executes operations to activate management capabilities for the correct delivery of network services. The SMP's interfaces are not visible by the user, but they realize the interaction between network functions and embedded management capabilities. Differently from traditional management, SMPs enable a high degree of automation and reliability in service provisioning, because they resolve and activate the required management functions in a distributed manner. The interaction between network functions and embedded management capabilities is realized through the model of embedded capabilities and their following implementation into functional components.
- **External Management Point (EMP):** It receives those operations that cannot be supported through embedded management capabilities, but which are needed for a complete management architecture. More specifically, the EMP includes i) operations that are out of the scope of self-management (e.g. report of physical failures) and ii) operations that do not perform efficiently through distributed algorithms (e.g. bulk information, like authorization repositories). The EMP is a useful instrument to make INM interoperable within the 4WARD overall architecture. For example, the report of a node failure can be forwarded to an interface of the Virtual Manager (WP3). Another example are requests of users' data from a repository, which can be instantiated via WP6's Network of Information (NetInf) API.

Self-management is initiated and maintained within the cloud delimited by the management points (cf. Figure 6-3). The GMP provides the instruments to define services and make them accessible to users. Additionally, it supports an operator in the maintenance operations (e.g. capacity planning and network extensions, etc.), which are performed today on a periodic and mostly manual basis. The SMP activates the processes for correct service provisioning, like

monitoring and self-healing functions. This occurs in the form of local and distributed control loops that are composed through the discovery mechanisms explained in Section 3.5.2.

The architecture so defined has the benefit of considerably simplifying management operations in the overall 4WARD architecture. This stems from the abstraction level provided by the GMP and by the underlying design of the distributed management capabilities. Additionally, the self-organization mechanisms discussed in the remainder of the document, imply a limited flow of feedback from the network to the operator in the form of violation of global objectives and aggregate metrics.

3.2 Instruments for a Clean Slate Design of In-Network Management

The need for basic design principles stems from the adoption of a clean slate approach within the 4WARD project. They are used as common ground for the design of the fundamental elements of this deliverable and for the successive consolidation in the next phases.

3.2.1 INM Principles

Following the method above, the INM framework stipulates five fundamental principles that will guide the design of management capabilities in the future Internet. Additionally, the INM framework combines technical results with a methodology for a gradual, non-disruptive, adoption of the novel INM functions. The first principle addresses see the basic ideas of the INM paradigm and captures all the potential developments of self-management features:

- **Intrinsic principle:** Management is intrinsic to the network. This fundamental principle captures the fact that the network *is* the management entity at the same time. As such, this principle dictates all architectural considerations of the INM framework.

The following three principles are consequences of the intrinsic principle and support the clean slate design of embedded management functions in the future Internet. These principles are extremal cases that will be relaxed in subsequent practical considerations.

- **Inherent principle:** Management is an inherent part of network elements, protocols, and services. As such, management functions come an inseparable, i.e. co-designed, part of the network. For instance, in a structured peer-to-peer network, overlay management is implemented inherently by the peer-to-peer machinery and can be considered a inherent management capability of it.
- **Autonomous principle:** Management is autonomous and does not involve any external manual intervention. This principle would leads to the adoption of a fully self-organizing management plane, which would also automate the enforcement of high-level business goals and physical intervention. This principle is clearly not feasible in its pure form, but it defines long term objectives for our research in automation of the future Internet: self-management functions will go beyond mere adaptation of device parameters and will strive towards the inclusion of domains traditionally excluded, like the management of business objectives (in collaboration with WP1) and the deployment of services through virtual resources (developed by WP3).
- **Abstraction principle:** External management operations occur on the highest possible level of abstraction. In the theoretical extreme case, the network may be triggered by an external stimulus only once at the beginning of its lifetime. All subsequent management actions and processes are concealed and autonomous in the sense of the autonomous principle. This principle guides us towards the definition of management interfaces for operators that hide internal self-management processes more than today's approaches.

Furthermore, the INM framework defines the following architectural principle that addresses the gradual architectural design methodology:

- **Transition principle:** The architectural design principles 2-4 should be implemented and developed in operative networks in a way that they can be gradually adopted. This principle is essential in that it allows gradual deployment of self-management 4WARD technologies and, in particular, assures marketability of INM results.

3.2.2 INM Transitional Degrees

While the architectural principles 2-4 are theoretic in nature, the transition principle breaks them down into a corresponding functional design space (shown in Figure 3-3). This principle supports a gradual adoption of these principles to various and practical degrees. In the centre of the disk, INM designates the extremal case where principles 2-4 are adopted in their pure form. The naming of principle 5 stems from the fact that between the intermediary steps on each dimension, a transition occurs to adopt a design closer to INM.

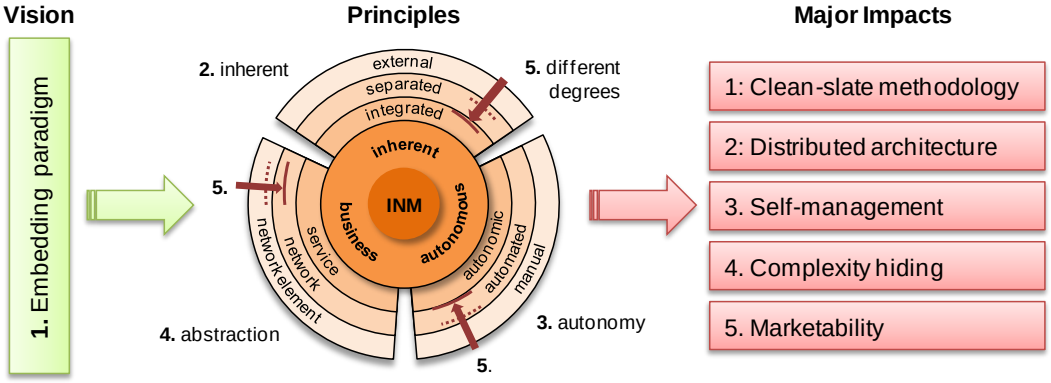


Figure 3-3: Transitional disk: three dimensions of the functional design space

Along the **degree of embedding**, the INM framework provides scope for a relaxation of the inherent principle. Management processes can be implemented as external, separated, integrated, or inherent management capabilities of the network. Integrated is weaker in that instead of indistinguishable management functionality, it designates visible and modular management capabilities, but which are still closely related to and integrated with specific services. Separated management processes are those that are more decoupled from the service, and include, for example, weakly distributed management approaches. External management processes include traditional management paradigms widely used today.

Along the **degree of autonomy**, the INM architecture allows for different degrees of autonomous management, from manual to fully autonomous processes. Manual refers to the direct manual manipulation of management parameters, such as manual routing configurations. Automated management can be typically found in the application of management scripts. Autonomic and autonomous degrees include intelligence that allows the system to govern its own behaviour in terms of network management.

Along the **degree of abstraction**, different levels of management according to the telecommunications management network (TMN) functional hierarchy [17] can be adopted. This dimension leads to a reduction in the amount of external management interactions, which is key to the minimization of manual interaction and the sustaining of manageability of large networked systems. Specifically, this dimension can be understood as moving from a *managed object* paradigm to one of *management by objective*.

As suggested by Figure 3-3, different parts of the network may adopt their specific degree of embedding, autonomy, and abstraction, based on practicability, specific goals and requirements. At the same time, the INM principles proactively support the transition in the functional dimensions in a technological aspect. If design issues are considered at the design time of new components, those components may encapsulate existing management functionality in a way that allows for a non disruptive transition to purer cases of INM.

3.2.3 INM Artefacts

With close attention to the INM principles, the INM framework defines three key INM artefacts that provide the elements by which the INM framework is realized (cf. Figure 3-4): the architectural elements, the INM deployment environment, and the algorithms. The architectural elements (cf. Section 3.3), comprise a number of components which are the building blocks from which INM functionality is constructed, including management capabilities, functional components, and INM entities. The INM deployment environment (cf. Section 3.4) facilities the instantiation, deployment, and operation of the architectural elements within the self-organizing management plane. The algorithms (cf. Chapters 4 and 5), which provide, e.g., situation-awareness, anomaly detection and self-adaptation, use the architectural elements and their interaction to realize their solutions and then deploy within the INM deployment environment. All three artefacts are pivotal to enabling INM and will be discussed in detail in the rest of the document.

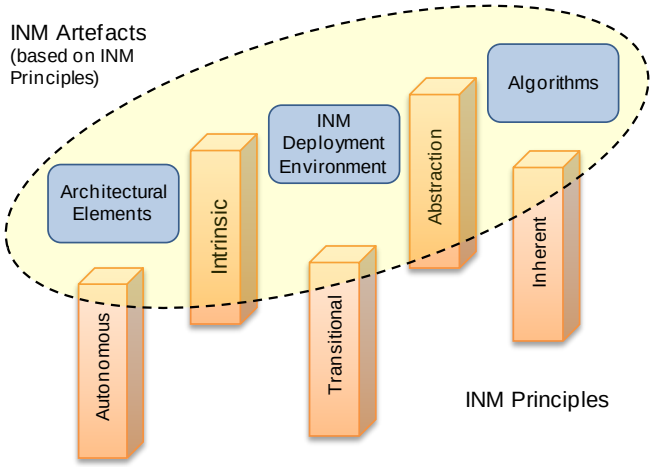


Figure 3-4: INM artefacts based on INM principles

3.3 Architectural Elements of In-Network Management

As first artefact, the INM framework introduces five basic architectural elements: management capabilities, functional components, INM entities, the INM protocol, and the INM registry. The architectural elements are found on two levels: the communication level and the implementation level (cf. Figure 3-5). The motivation behind this distinction is to show how the design and implementation of a management process can be separated from a management process that is being executed within the self-organizing management plane. For both the design/implementation and execution of a management process, several players with potentially different roles are involved (cf. Section 3.1).

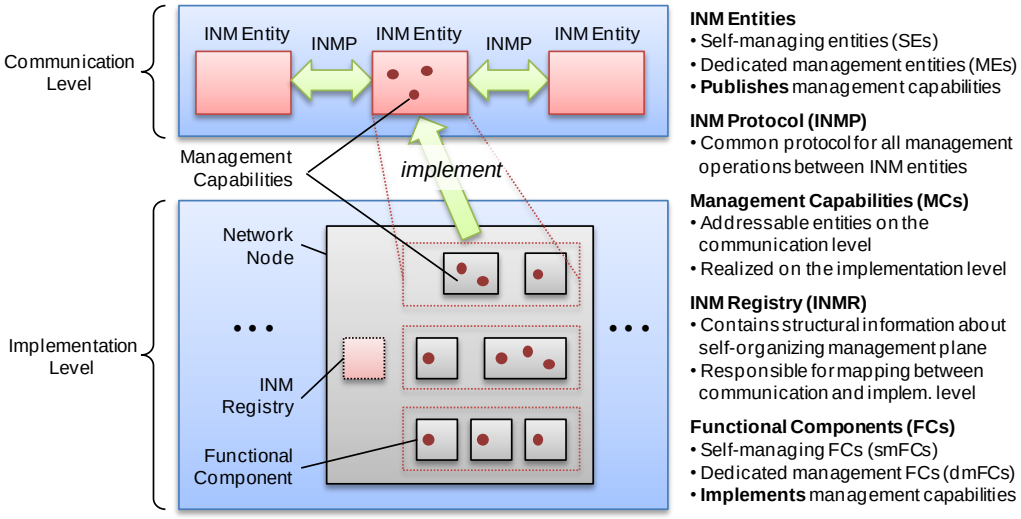


Figure 3-5: Overview of the architectural elements of in-network management

On the communication level, INM defines the concept of *INM entities*, which are, in a nutshell, the entities visible in an operating network and between which network management tasks are

performed. INM entities are distinguished into two types. *Self-managing entities* (SEs) are the logical constructs which contain service and management functionality, whereas those INM entities that contain only supporting management functionality are referred to as *dedicated management entities* (MEs). Between INM entities, all management operations are mediated via the *INM Protocol* (INMP), which carries messages for collaboration or organization tasks to be performed between INM entities. Typical examples for operations carried via INMP relate to synchronization, coordination, or any algorithm-specific management task.

On the implementation level, INM provides functional components (FCs), which reflect an implementation-specific view on INM and which are defined in analogy to INM entities. Hence, two types of FCs are distinguished: *self-managing functional components* (smFCs) and *dedicated management functional components* (dmFCs). In short, a combination of functional components implements an INM entity, indicated by the *implement* arrow in Figure 3-5.

M*anagement capabilities* (MCs) are fine-granular elements of management functionality from which more complex management functions can be constructed. MCs are designed using object-oriented software engineering approaches. MCs apply to both the communication and implementation level, and the INM Registry (INMR) provides the means to translate between both levels. On the communication level, MCs can be addressed between INM entities in order to realize MC interaction, and thus, form more complex management processes. On the implementation level, MCs are realized by a specific implementation. By grouping management capabilities of one or more FCs together, MCs propagate their interfaces outwards such that they become visible through INM entities on the communication level.

3.3.1 Perspectives

The full deployment cycle of management capabilities requires different expertises and technical knowledge of the architectural elements; in practice, management capabilities can be co-designed with network functions only if the perspectives of different actors are taken in account. We identify three roles with a specific perspective on the INM framework:

- The **network architect** is one who builds the networks and its services. He/she does not modify any of the architectural elements, only selects and uses them;
- The **developer** is a person who wishes to design specific management functionality, e.g. an algorithm developer;
- The **implementer** is the one who will design and develop some service complete with management functionality, e.g. the designer of a generic path (GP).

Specifically, these roles are in accordance with the APC deployment framework of WP2 and the Vnet business models of WP3. Furthermore, the definition of these roles also supports the “gazelle” scenario envisioned by WP1 (cf. D-1.1 [2]): considering the “virtualization” use case of D-1.1, the different roles (provider, operator, service offers, users) can be make use of different instruments of INM, according to their perspectives.

Table 3-1 shows the relationship between the different roles and the architectural elements. This implies that the network architect must understand what the INM entity and the MCs are and how they should be used. The ‘network architect needs not to have a full understanding of the functional components and the INM Protocol. This table can guide different actors in the future Internet for the assignment of responsibilities and design perspectives.

Architectural Element ▶ **▼ Role**	Management Capability	Functional Component	INM Entity	INM Protocol	INM Registry
Network Architect	X		X		
Developer	X			X	
Implementer	X	X	X	X	X

Table 3-1: Relationship between roles and architectural elements

3.3.2 Management Capabilities

Management capabilities (MCs) are the fine-granular architectural elements implementing a specific function: they can be addressed and can be composed to construct complex management functions (e.g. performance monitoring, situation awareness). In conjunction with INM entities (cf. Section 3.3.4), two different interfaces are defined via which INM Entities publish their management capabilities: organization and collaboration interface. In order for MCs to support this distinction, each MC's published method that can be called on the MC's functionality is mapped to either one of these two interfaces. The left side of Figure 3-6 illustrates the various forms that an MC's interface can take.

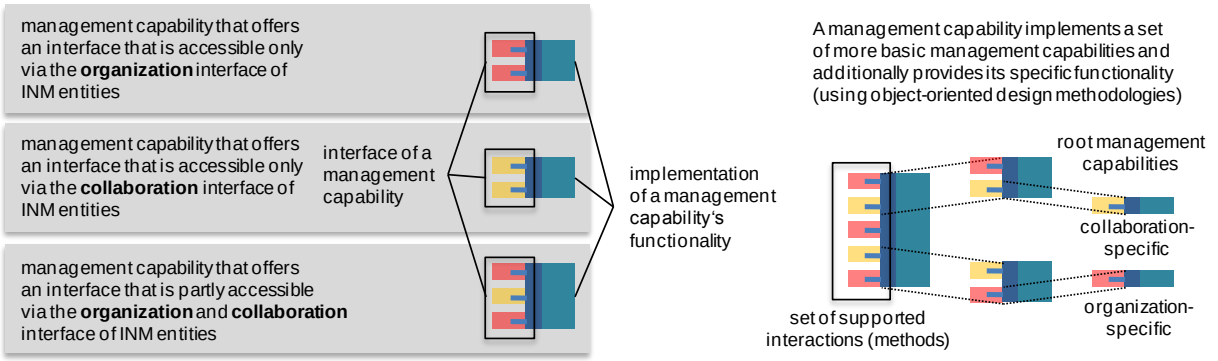


Figure 3-6: Basic model of management capabilities

The design and implementation of management capabilities follows an object-oriented design approach. Therefore, known object-oriented methodologies (e.g. design patterns) are directly applicable to MCs. The right side of Figure 3-6 illustrates the construction of a more complex management capability (e.g. a monitoring capability) from a set of more basic MCs. Each MC specifies whether each of its methods is published via the organization or collaboration interface of an INM entity (cf. also Section 3.3.5).

The INM framework defines MCs in a way that they can reside at any of the degrees of embedding introduced in Section 3.2.2. Relative to INM entities (specifically, self-managing entities, SEs), hence, MCs take the following forms:

- Inherent MCs are very closely tied to the service of an SE, meaning that they are co-designed and implemented together with that service. They do not offer visibility to the management interfaces of the SE. The closest similar approach of today's networks is represented by integrated OAM functions of Ethernet standards.
- Integrated MCs are functionally related to a specific service of a SE, but their functions are offered via interfaces for linking to other management operations..
- Separated MCs are decoupled from the service of an SE and are used for general management operations, like aggregation of network-wide performance indicators.
- External MCs reside outside the scope covered by the INM paradigm and capture management functionality that is external to a communication network's elements.

MCs should be designed such that they can potentially run at any level of embedding, according to the principles of Section 3.2.1. The implementation of inherent and integrated MCs provides an instantiation of the SMP introduced in Section 3.1 and therefore realizes the interaction between embedded MCs and network functions. What level of embedding an MC resides in is at the discretion of a developer/implementer. This also results in the potential to create an external library of MCs which could be queried and used by SE developers.

Figure 3-7 provides the initial model of a management capability. An MC has a main interface which provides generic capability-specific methods (e.g. getInfo). This generic interface can be extended by a capability-specific interface; the model shows as examples performance, routing and configuration capabilities. These extension interfaces provide the specifics for each capability type. The model also shows a SpecificDeviceCapability interface extending the

main interface, which highlights the potential to extend the model to also address the likes of device capabilities and others. Each specific capability interface has its own implementation class (shown by the PerfCapaImpl example). For each capability, there is also a CapabilityAbstractDescriptor a class representing the generic description of an MC. This class is extended also by a specific descriptor, e.g. PerformanceDescriptor in the model. The model is designed to give scope to the designer but also to provide a structure which must be adhered to when developing management capabilities.

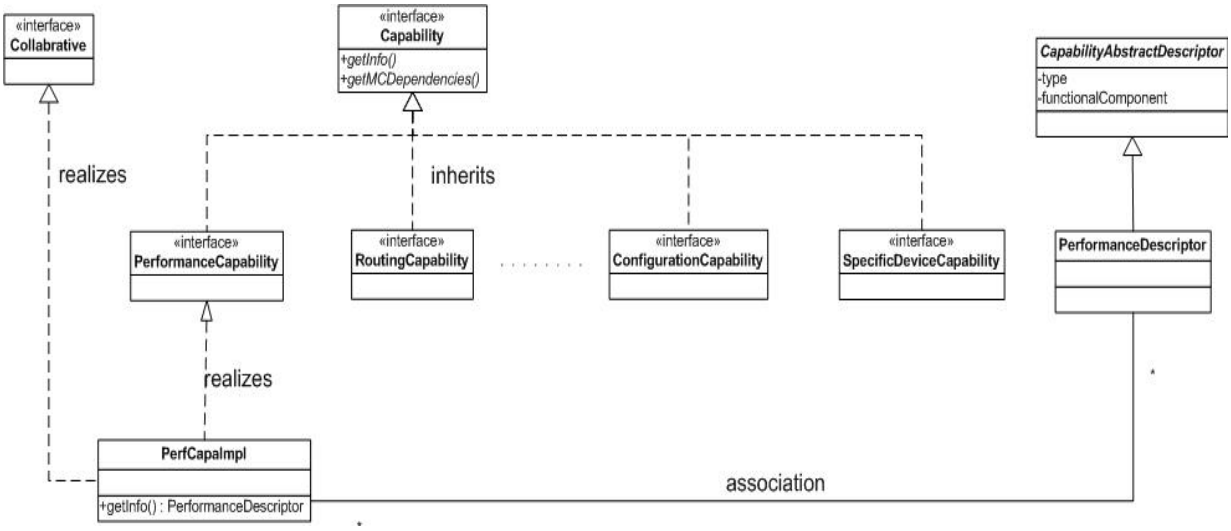


Figure 3-7: Management capabilities UML model

3.3.3 Functional Components

Functional components (FCs) are the basic implementation-specific elements on a network node that can encompass both management and service functionality in one entity. An FC might represent, for instance, a protocol (sub)layer or any software module that encapsulates a specific service or part thereof. FCs are distinguished into the two types *self-managing* (smFC, left side of Figure 3-8) and *dedicated management FC* (dmFC, right side of Figure 3-8). The distinction is motivated by the fact that certain management functionality is specific to a service (e.g. an smFC dealing with routing performance), while others is generic and may be used by several other FCs (e.g. a dmFC implementing a cross-layer neighbour table).

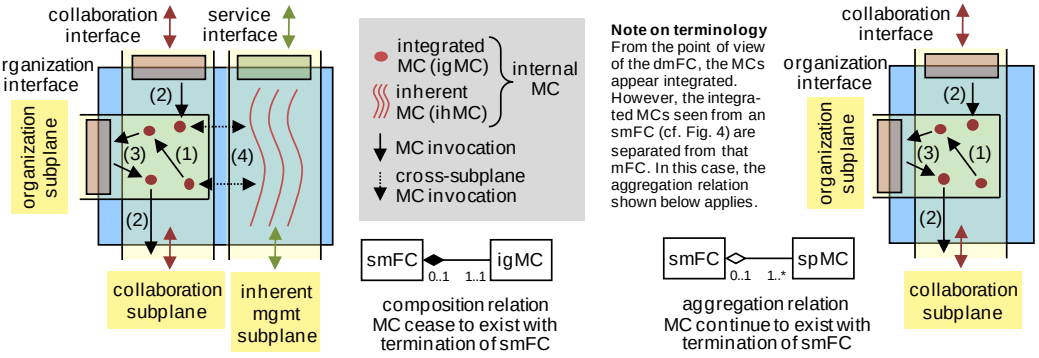


Figure 3-8: Functional components

A self-managing functional component offers its service via the service interface (e.g. the sending of frames in a MAC module) and provides two additional interfaces that enable it to communicate with either organization-specific or collaboration-specific management peers, which is in analogy to the interfaces of the INM entities (cf. Section 3.3.4 and 3.3.5 for INM entities and their relation to SEs and FCs). The collaboration interface is for any collaboration between FCs in order to access one another's MCs so distributed management objectives can be achieved collaboratively. The organization interface is related to governance and other

high-level tasks and mediates between external (business and technical management) and internal (both integrated and inherent) management.

On the right side of Figure 3-8 we show the structure of a dedicated management FC. The difference from smFCs is that the dmFC lacks a service and is limited to performing management-specific tasks only. Since a dmFC's management capabilities may be reused by several smFCs, this type of FC contains only integrated management capabilities. At this point we are able to identify the degree of separated management. When considering the management capabilities of a dmFC, they appear separated from the smFC if they are used by that smFC. This degree of embedding is key in providing a smooth migration of management functionality from external systems (e.g. management stations) closer to the relevant self-managing FC. Figure 3-8 includes also a view on the distinction between integrated and separated management in a UML style notation. In this view, integrated MCs (igMC) of an smFC can be considered to follow a composition relation (left side of Figure 3-8), whereas MCs that are separated from an smFC (abbreviated spMCs) and contained within a dmFC match an aggregation relation (right side of Figure 3-8).

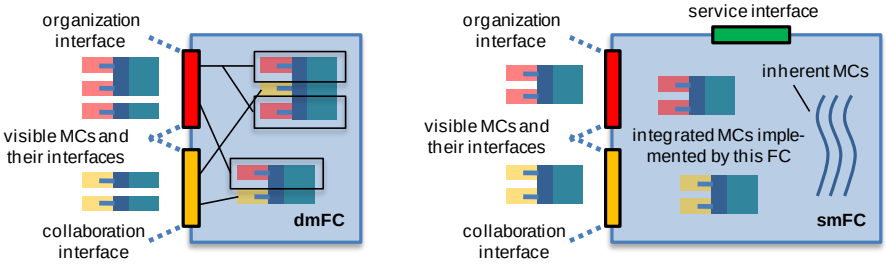


Figure 3-9: Relation between functional components and management capabilities

The relation between MCs and FCs is depicted in more detail in Figure 3-9. Independent of the type of FC (dmFC or smFC), the methods used to access MCs are mapped to the corresponding organization and collaboration interface of an FC. Considering a single interface, what is visible at that interface is, hence, a set of MCs with the subset of methods that possess the visibility of the specific interface. MCs offering methods only to one of the organization or collaboration interface are consequently visible only at that interface.

3.3.4 Self-Managing Entities

Self-managing entities (SEs) are the logical constructs that encompass the properties necessary to achieve autonomous operation of the network infrastructure. An SE is thus built around a set of principles, demonstrates a set of properties and communicates through a set of interfaces. Self-managing entities are organised (either through a predefined order, or via self-organisation) in a hierarchy according to the relationships between services.

In INM, a "self-management by objective" paradigm replaces the "managed objects" paradigm of traditional network management. Service-related goals are received from a business service manager (human or software application) by the highest level of SEs and then refined down through the hierarchy and propagated north-south via the organisation interface. This interface can provide an instantiation of the GMP introduced in Section 3.1 and therefore be used a provider to inject objectives and SLA into the network. The collaboration interface enables east-west peer-wise cooperation between SEs located on the same level of the hierarchy. Service access points enable the delivery of services to users.

The organisation interface includes all the protocols and operations necessary for composition along with capability level statements (CLS) and service level agreements (SLA) regarding the services provided by the SE. The collaboration interface performs the peer-wise interactions that were setup over the organisation interface in order to achieve the service delivery goals. MCs are exposed over both interfaces in direct relation with the respective functionality.

Principles of Self-Managing Entities

The major principle SE entities need to adhere is the intrinsic principle detailed in Section 3.2.1. The other principles are derived from good practices well established in software engineering: modularity, extendibility, interoperability.

Modularity principle: The modularity principle states that responsibility for achieving different goals is delegated to different components of a system. A degree of separation is thus achieved based on functionality and delegated responsibility. This is an adaptation of software engineering principles expressed as early as 0. The modularity principle is an enabler for the re-use of existing SEs to produce different end-user services through multiple possibilities of combining basic or complex modules.

Interoperability principle: In software engineering, interoperability is defined as the ability for multiple software components written in different programming languages to communicate and interact with one another [38]. In the context of SEs, the interoperability principle pledges support for key standards and broad compatibility. As a result, network equipments and software manufactured by different vendors will be capable of interoperating to provide basic (cf. also extendibility principle) self-management properties.

Extendibility principle: The extendibility principle enables designing SEs in a manner that is open to changes and improvements. A hierarchy of SEs becomes thus open to provide value-added services and enhanced functionality. Interoperability is not always necessary for these extended services and functions. As a result, vendors and operators can provide rich value-added services to promote their products and protect business interests.

Properties of Self-Managing Entities

The properties of SEs are an important cornerstone of the INM framework allowing the replacement of the current “manager-agent” based architecture with a more agile, widely distributed construction. Kephard and Chess [39] formulated a set of four aspects to be considered for achieving self-management in an autonomic computing scenario. We believe that the reorganisation and addition of new properties (when compared to the self-management aspects of [39]) would enable a wide-scale decentralisation of network management functionality. The INM framework distinguishes the following properties of SEs: self-knowledge, self-management, self-protection, composability, auditability.

Self-Knowledge property: Self-knowledge allows the SE to understand its identity, current status and state as well as its role in the system. It builds upon the following functionalities:

- *Identity and trust:* Each SE has to be equipped with an identity. The uniqueness of the identity would result out of the wider 4WARD discussions on naming and addressing. The identity can be authenticated as originating from a trusted source. Exact details remain to be studied, but a public key-based solution along with tamper-resistant hardware modules in the network elements could be the basis for identity at the device level. In order to verify an identity, a common trust anchor must exist between the authenticator and SE to be authenticated. Keys could be used to verify an identity. However, in order to do that, some process is needed whereby a specific identity is tied to a specific key. A similar approach was suggested as part of the 4D architecture by Greenberg et al [40].
- *Self-description:* Each SE can describe its capabilities, the quality of service associated and all parameters associated with the interfaces it provides to other SEs. The self-description property allows external tools to interact with the SEs when composition of services is necessary. It also acts as an enabler for self-organisation of the SE hierarchy.
- *Self-monitoring* of all internal functionality and interfaces: The SE observes continuously the parameters related to its objectives in order to evaluate the level of fulfilment. Deviations from target values are reported to the SE that uses the service.

On a given level of the hierarchy, distributed monitoring algorithms would need to be employed in order to satisfy this property.

Self-Management property: The self-management property provides the methods for interacting with other SEs in a way that allows maintaining the overall state within certain objectives. The following categories of functionality have been identified as part of the self-management property:

- *Self-configuration* provides an SE with the ability to bootstrap itself, infer or determine the correct values of all the configuration parameters (perhaps with the exception of a minimum set of parameters related to the identity, which would need to be provided externally).
- *Self-diagnosing* enables the SE to perform functions related to fault prediction, identification and correction, both at the individual level within an SE and at the system level in cooperation with other SEs. Distributed root cause analysis and anomaly detection are example of algorithms that would be needed in order to fulfil this functionality.
- *Self-optimisation* allows the SE to take tactical decisions related to configuration parameters and influence the topology of relationships with the SEs assembled through composition and also the associations with other SEs in order to optimise the system as a whole.

Composability property: In order to be able to create new services based on existing ones, SEs should be able to aggregate for producing composite services. SEs should have a standard set of interfaces and protocols so that they can be composed to produce composite services. The self-description, part of the self-* property and the modularity principle also facilitate composability. The main aspects of interest for the SEs in relation to the composability property are how to address aggregation and desegregation of the control loops associated with self-management properties of SEs placed at different levels of the hierarchy.

Auditability property: The auditability property enables SEs to be verified on the fulfilment of their service as well as on their operating state. The self-monitoring property is an enabler for the accountability part of the auditing process. Auditability allows an authorised third-party to verify the values reported by the SE. Visualisation tools may use the auditability property to obtain values of parameters internal to the SE. This type of functionality would be particularly important for increasing the confidence of the operator on the routine use of network management techniques with high degrees of autonomy.

3.3.5 Relations and Basic Interworking between Architectural Elements

In Section 3.3.3 we have considered the relation between management capabilities and functional components. In this section we briefly cover the remaining relations between functional components and INM entities and how these architectural elements interwork with support of the INM protocol and INM registry. Figure 3-10 depicts the relation between functional components and INM entities and is a detailing of the relation indicated in Figure 3-5 by the *implement* arrow.

The dashed lines inside of the hybrid module (left side of Figure 3-10) and the IPv6 module (right side of Figure 3-10) shows exemplarily the functionality that is encompassed by a specific SE. In the example, the SE extracts the IPv6-specific functionality that is typically comprised by a TCP/IP protocol suite. Considering the single hybrid functional component, each of the organization and collaboration interface publishes the complete set of MCs supported by this FC (for ease of exposition, the capabilities are not displayed in the figure). The corresponding extracted SE contains, therefore, only a subset of the MCs exposed through the FCs' management interfaces. On the right side of Figure 3-10, the interfaces of the SE map one-to-one onto the interfaces of the corresponding FC, because the way services are modularized in the example corresponds to the SE. In both configurations of Figure 3-10, the self-managing entity is the communication-level construct that is implemented

by any sensible modularization in the form of FCs on the implementation level. In principle, it is possible to implement an SE by any combination of FCs or their parts, even from different nodes, as long as the properties of the SE according to Section 3.3.4 are satisfied.

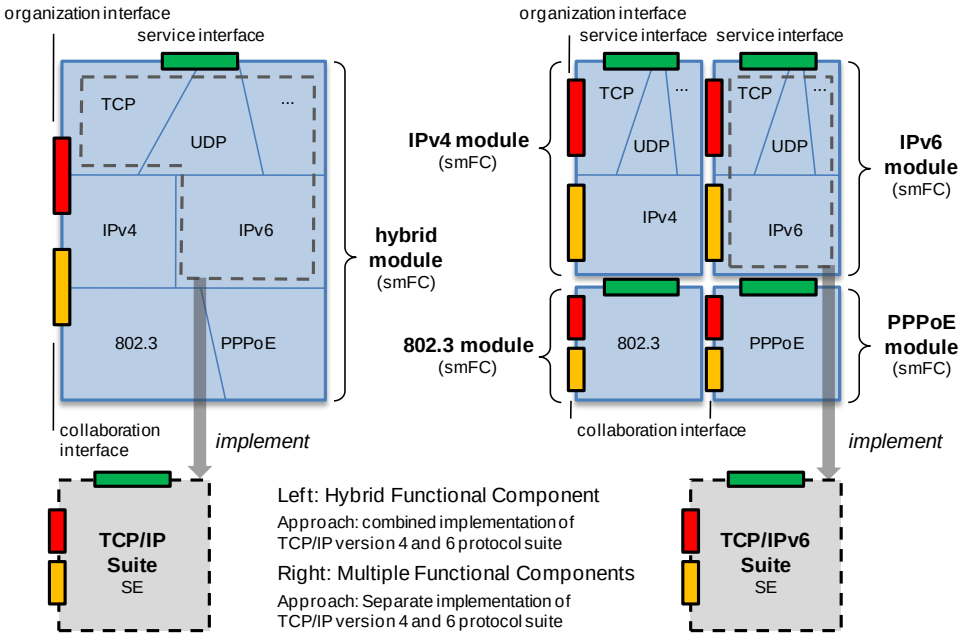


Figure 3-10: Relation between functional components and INM entities

Figure 3-11 shows how SEs and FCs interwork on the level of network nodes. The figure illustrates, in particular, the role of the INM registry (INMR) and the INM protocol (INMP). The figure also shows the case where a dedicated management FC and a self-managing FC make up an SE (the two FCs located at the bottom right of Figure 3-11). The INM registry as it is shown in Figure 3-11 is only a logical depiction of the functionality required to translate between the communication and implementation level. Furthermore, it is realized by using MCs, FCs, and SEs. For example, one way of providing the INM registry is to design it as a single MC (e.g. “InmRegistryCapability”), which is contained, possibly together with other MCs, within a single dedicated management FC that in turn realizes the dedicated management entity of a node. Another way of providing INM registry functionality is to embed its functionality in the form of one or more MCs within each of the SEs of a node, thereby forming an embedded and distributed INM registry.

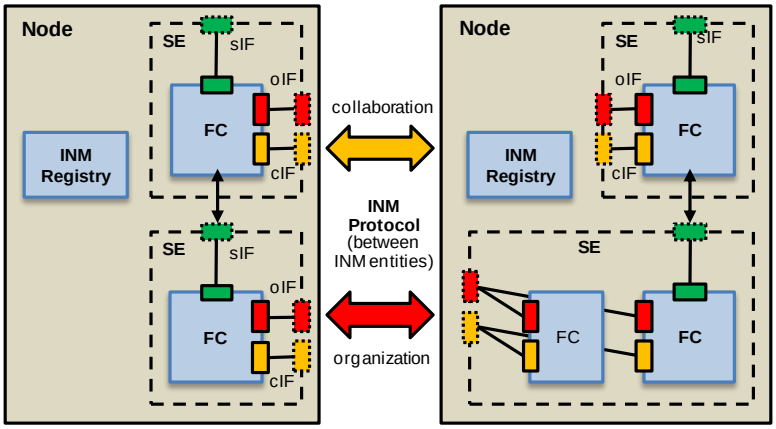


Figure 3-11: Node-level view of SEs, FCs, the INM protocol and registry

Apart from the relations discussed in this section, we will address in more detail how the interfaces of management capabilities are propagated up to the management interfaces of self-managing entities in the context of the lifecycle of INM entities in Section 3.5.1.

3.4 INM Deployment Environment

The architectural elements of INM have been defined. These elements must have an environment to run in. One possible model of such an environment has been designed which supports the INM architecture. This environment must be modular, in that the most fundamental components or artefacts can be easily exposed and also that the more high-level components can be easily added and extended. Figure 3-12 shows the components which make up the environment. The environment has three main layers, the INM applications, the INM kernel and the underlying platform layer.

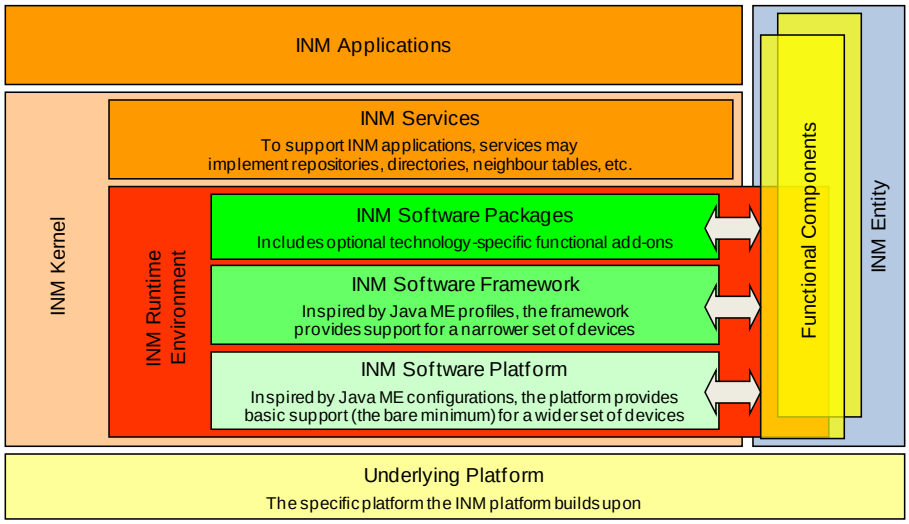


Figure 3-12: INM deployment environment

The INM applications layer is an area where management applications can be deployed. These applications have the ability to interact with instances of the architectural elements and also with the INM kernel, once they have the correct permissions. The INM kernel is a privileged area within the environment where components or services are protected from application interference, in that access to artefacts inside of the INM kernel is restricted. Security measures are enforced when accessing this area of the environment. INM services are utilities whose primary task is to provide fundamental support for INM functionality. The developed INM services will be relative to the capabilities of the node itself and to the features which are supported by the INM runtime environment.

The INM runtime environment is a container in which FCs and services can execute. A specific runtime environment can be extremely light-weight or more heavy-weight, e.g. similar to the Java runtime environments. The INM runtime environment is actually a concrete implementation of an execution environment which itself is a formal specification of an environment for applications and services. A runtime environment may contain a virtual machine (e.g. JRE and Microsoft .NET Common Language Runtime) or may contain only libraries (e.g. DirectX and GTK+). The INM runtime environment contains three layers or levels of abstraction: the INM platform, the INM framework and the INM packages.

The INM platform contains fundamental libraries and capabilities. It provides interfaces to any FC and the basic communication between them and a Functional Component Abstraction Layer to provide technology-independent access to resources and FCs. The INM framework provides support for ease of implementation of INM services and applications. It will contain libraries, capabilities and utilities applicable to a narrower set of devices. Example functionality which may be supported by the framework layer: command mediation: a way to send commands to functional components and receive responses; event handlers: handlers which can process incoming events from functional components or lower level components; event filters: filters which can be tailored and then applied to generated events; management of properties: used to set functional component and service level properties. The INM packages will contain optional libraries, capabilities and utilities supported by specific types of devices.

FCs and MCs span the application and kernel layers in that they can interact and collaborate at any level. The INM entity is shown in this diagram just as an aid to understanding, because in the actual deployment environment, FCs will be the container which will be visible and the FC (or numerous FCs) realize the logical INM entity.

3.5 Self-Organization of INM Processes

The architectural elements introduced in the previous sections provide the modular instruments to perform management operations through a distributed approach. This section explains the self-organizing mechanisms to activate and compose the distributed management capabilities to build management processes.

A distributed architecture requires a set of technical instruments to compose functions located on different locations and at different scales. The structuring of large networks has been mostly a non-automatic task in the past; the planning phase of networks (which includes defining domain substructures) is performed prior to any rollout.

3.5.1 Lifecycle of Self-Managing Entities

We first describe briefly the complete lifecycle of management capabilities. For that, we put the self-managing entity into the centre of our considerations, which contains a service and the corresponding management functionality. In the following, we describe the lifecycle of a self-managing entity along four phases: design and implementation, deployment and activation, operation and reconfiguration, and termination and deregistration.

Design and Implementation

Initially, an SE is designed with a communication-level view in mind where no reference to functional components is required. The modelling of MCs follows the descriptions in Section 3.3 and part of them are to be published via the SE's organization or collaboration interface. To support self-organization, the SE should specify dependencies on two levels: dependencies to MCs, implicitly given by the MCs included by the SE, and dependencies to SEs, specified by the designed SE in order to connect to existing types of SEs for composition, like described previously in Section 3.3.4.

After the design of an SE, it is implemented in the form of FCs. Focusing on the management part, each of the previously specified MCs needs to be provided by the implementation. This can be accomplished either by a concrete implementation of the MC (especially in the case of inherent and integrated MCs) or by the specification of a dependency to be resolved during the activation phase (especially in the case of separated and also integrated MCs).

Deployment and Activation

The deployment and activation of an SE may occur by introducing a new network element into an existing physical network architecture, by creating a virtual node in a virtual network and populating that node with service and management functionality, or, in general, by creating a network based on the architecture principles and concepts (APC) developed in WP2.

The activation of an SE is performed through a plug-and-play process, where the discovery mechanism is used to resolve the required dependencies described in the previous phase. Once all discovery steps have completed successfully, the self-managing entity can be activated. At this time, the SE registers with the INM registry (INMR). Depending on the realization of the INMR, information about the activated SE may be stored in a node-local INMR or in several INM registries on different (physical or virtual) nodes. The final step is activating the SE, where the management processes start to execute.

Operation and Reconfiguration

After the activation of an SE is complete, it begins operating within management control cycles that are active within the self-organizing management plane. According to Figure 3-5 and Figure 3-11, the INM protocol (INMP) provides general primitives for interactions between MCs. At runtime, the INM deployment environment described in Section 3.4 will map the INMP

to remote calls between nodes or local function invocation (in conjunction with generic paths, developed in WP5, which provide advanced transportation services).

In the simplest case, the SE performs management tasks that are triggered by various actions. For example, a service request may trigger actions in the management plane by the coupling between service side and management side that is defined via a subset of the SE management capabilities. A typical case is the reporting of an event, a mechanism described in more detail in Section 5.2.5. Another example is the triggering of a management function by invocation of the SE's capability from another SE's capability. A very typical interaction of this kind is a distributed real-time aggregation process that is set up between different SE's, aggregation being defined in detail in Section 4.1. Yet another example is the triggering of a management action via one of the management points defined in Section 3.1. In general, the possible ways of triggering management functions of the considered SE are manifold and strongly depend on the purpose and task of the specific SE, in particular, the specific set of management capabilities that the SE contains.

To address mobility aspects, dynamic re-discovery and re-registration of management capabilities at runtime and possibly with other self-managing entities becomes necessary. This fact emphasizes especially the need for discovery mechanisms that are suitable for embedded management processes according to the INM paradigm, and which are therefore discussed in more detail in Section 3.5.2. Additionally, the technical challenges envisioned by certain scenarios, such as the scenario 2 large-scale adaptation and the scenario 4 response to dramatic events described in deliverable D4.1, are addressed by the self-organizing functions of the INM Entities, such as the (temporary) suspension and restoration MCs according to network conditions and service requests. Such high-level behaviour is currently being considered and will be elaborated in more detail in the following documents.

Termination and Deregistration

The last phase of self-organization of management capabilities is their termination and deregistration. This occurs, for instance, when a service is revoked. As a consequence, the associated management functions must be released in an ordered manner, in other words as a reverse execution of steps that occur during deployment and activation (phase 2). Termination can be divided into three main steps: The first step is the resolution of dependencies with other SEs. As a result, the SE may be either terminated or it remains integrated in the self-organizing management plane. Eventually, it might be possible that the SE, while being tied to other SEs, might be replaceable by other functionality.

Once it is determined that the SE can be removed, its termination is initiated. For that, its individual MCs need to be consistently terminated. To illustrate the deactivation, consider, as an example, real-time monitoring (cf. Chapter 4). In this case, an aggregation tree may be formed between several SEs. Because aggregation will continue to function with one less SE (that is, one less node in the aggregation tree), the SE can be removed and the aggregation tree can be reconfigured according to the rules that the aggregation algorithm imposes on the tree. The example of aggregation illustrates the deactivation of only a single management capability that models aggregation. After the termination, the SE is finally deregistered from the INM registry. Again, depending on the realization of the INM registry, this step might imply actions on one or several (physical or virtual) nodes.

3.5.2 Discovery

As we have seen, the resolution of dependencies is essential during the various phases of the lifecycle of self-managing entities. For that reason, suitable discovery mechanisms that enable the deployment of management functions in a distributed way are indispensable for INM in the first place. The discovery mechanism we address consists in the discovery of management capabilities between functional components that can be located on the same physical node or different nodes. A discovery request may originate in the following situations:

- A service is set up and the corresponding management processes have to be started and configured as well (resolution of dependencies between services);
- A management process is running and it needs to collaborate with other management capabilities (resolution of dependencies between management capabilities).

The output of the discovery mechanism is a handler to one or more management capabilities matching the characteristics specified in the request. The implementation of a handler depends on the reciprocal location of the MCs and it can be mapped to a locator internal to the INM entity itself or to a GP endpoint.

Since we are separating the specific mechanism to discover MCs from the description language to describe them, the proposal for discovery here discussed can be adopted for the service discovery in Vnets. In this case, the XML request specific to the Vnet description can be seen at the same level as description of MCs.

Functional Issues

While traditional approaches to network management make little use of discovery functions, a number of existing discovery protocols have been developed mainly for service discovery purposes. Typically the output of the process is the address of the service provider for the requested service. Such discovery methods can be mapped to MC discovery if we regard MCs as “management services” provided by the FCs in the network.

Hereafter, we consider several functional issues that are addressed differently in the existing service discovery protocols and try to extract the requirements that apply to the discovery needed in INM. The clean slate design gives the opportunity to build on those a more general and flexible discovery mechanism for management functions.

Storage of MCs description: While centralized storage approaches assume a single server holding all information, distributed structured and unstructured approaches make use of routing-based or broadcast/multicast-based discovery mechanisms, respectively. While centralized storage is not appropriate for a distributed INM architecture, distributed approaches may be suitable depending on the type and requirements of a particular network, such as network size and dynamics. Unstructured approaches seem particularly promising, as they do not require the overhead to build and maintain structures between nodes.

Local view and caches: When considering distributed storage approaches, a related issue is to define the view that each node must maintain. In some protocols, like Universal Plug-and-Play (UPnP), nodes maintain a full view of all services in the network. In other approaches, like the Group-based Service Discovery (GSD) protocol, nodes locally cache a limited view of service descriptions available within a diameter. This approach improves scalability and should be therefore taken into account in INM.

Description language: Many of the existing languages follow an attribute-value structure and can be mapped through XML encoding. We believe that for our purposes the description language is not a decisive factor, although it is needed to enable search mechanisms. An object oriented approach with value/parameter sets and encoded as XML is adopted.

Search method: search methods operate in close dependency with the type of storage used for discoverable information. Depending on how much structure is provided by the storage approach, search is conversely more efficient. Therefore, distributed structured approaches best support search efficiency by enabling routing-based querying. Moreover, both push and pull approaches are conceivable, where management capabilities are either proactively announced or reactively discovered. As in the case of storage, a suitable trade-off between searching methods depends on the characteristics of the considered network.

Matching rules: a diversity of matching schemes may be used to define when a discovery is successful, including attribute-value pair matching or wild-card matching. When more than one match occurs, additional algorithms are required to select and terminate a discovery process. The choice of matching rules and selection rules mostly depends on the requirements of the

querying entity. While a specific MC may be discovery by the first perfect match encountered, wild-card matching could be helpful when human intervention is needed.

Maintenance: When a service description changes, consistency may be maintained either by sending an advertisement with the new description or sending an event (publish/subscribe). Considering that dependencies and collaboration in INM are established between MCs, the publish/subscribe model seems to be suitable to send updates of the service descriptions to MCs that are interested in that. Two methods can be used to keep a consistent view of the available services: in a soft state model, services periodically send updates to confirm their availability; in a hard state model, records do not expire and need to be explicitly deleted when unavailable. For INM, hard state may be more appropriate due to rather stable records, as management functions are often performed in correspondence with a service.

Approach for a Discovery Solution for INM

While existing discovery protocols provides functionalities that can be reused for the discovery of management capabilities, they do not address the following issues:

- No mechanism for inter-domain discovery is provided. Existing protocols are designed to work within a domain or within a network with specific characteristics. This fact is an obstacle for the deployment of existing protocols to the INM architecture.
- No protocol can be mapped to different transport layers: a protocol can discover services that are provided by a specific layer but cannot address cross-layer aspects.

Moreover, scalability, robustness, and timeliness are essential requirements for discovery applied within the scope INM. An effective way to provide this set of requirements is to limit the scope inside of which discovery takes place, e.g., by imposing a diameter or domains within which searches are concealed. This approach may significantly reduce the cost of discovery and in turn allow for more redundancy and concurrency of discovery mechanisms. As a result, more robust and timely discovery becomes possible by relying, for example, on backup and parallel execution of discovery instances in situations where one mechanism might fail or require a performance boost with respect to timeliness.

Domains, in particular, provide a powerful way to group nodes with similar characteristics, e.g. their degree of mobility. Within a domain, a discovery protocol can then be chosen according to these characteristics and even adapted dynamically when the characteristics change. In order to interconnect domains in terms of discovery, strategic nodes would be in charge of translating discovery requests between different domains, for example, from the home network to the wide area network. Because networks may be designed and deployed by individual network architects, discovery mechanisms in several domains will likely differ, e.g. in the storage system or in the search method. A query that cannot be resolved in one domain could be forwarded to the other domain by the strategic node, which analyzes the query and convert it according to the format and requirements of the new domain.

In the INM architecture FCs are grouped into self-managing entities (SEs), which represent a higher level of abstraction for organizing FCs according to their properties and functions. Discovery mechanism could operate between SEs through the SE's collaboration interface whenever the management process needs to locate management capabilities that are contained in another SE. The communication protocol designed to exchange messages between SEs could include a discovery type of message; moreover, a discovery message could be used both to search for a management capability and also for another SE. One of the properties of a strategic node would be being able of bridging discovery messages between different domains. This could be a property of a particular SE, like an inter-domain SE.

Given that different discovery mechanisms can coexist in the network, it would be desirable to define a common interface or a set of discovery primitives, which could be extended in a more specific way by each discovery protocol; at the same time, this would simplify the translation of discovery messages between domains performed by strategic nodes.

3.5.3 Structures for INM

The design elements discussed in the previous sections build a highly distributed architecture, where MCs can be discovered and executed on a peer to peer basis. Network Management exercises control over a network and has tight relationships with the structure(s) of the network. The classical structure is to manage groups of devices using an element manager, while several of these are controlled by a domain manager, and on top of everything, the actual network management. The state of the art is represented by larger, umbrella-like NM systems that cover several smaller NM systems. NM is used to create or modify network structures, and mostly this involves human operators to perform the separation of node sets.

The construction of a structure among INM entities can be triggered by different requirements and an analysis of the scenarios in D-4.1 produced a list of such requirements. Analogous concepts are adopted in 4WARD, such as the GP compartments [5]; with this respect, the INM structures can be certainly mapped one to one to those concepts, but the structures can be enforced in a more loose way, allowing for example the establishment of inter-domain or inter-compartment optimizations. The criteria for structuring are the following:

- Physical possession (ownership) of subsets of nodes by different shareholders. This needs to be reflected and partially enforced by means of network management.
- There may be limiting factors to some INM algorithms that forbid their use beyond a certain number of nodes, or hops or geographical distance. This range can be interpreted as a structure, and, if this structure is fortified (i.e. if the network becomes aware of it e.g. by using it regularly), it becomes a network domain eventually.
- Some network management operations shall have limited scope as they are valid only for e.g. certain regions, access technologies, handsets or home networks.
- Topology (geographical or physical connectivity).
- Administrative issues, e.g. different priorities for different parts of an operator network
- Performance (e.g. the “location areas” in 3G).
- Service specific requirements.
- Virtual networks (Vnets, WP3) can be treated as individual domains.
- Compartments according to WP5 can be mapped into an INM structure.

Structures can separate, for example, routing areas from each other; they implement the „Intranet“ vs. „Internet“ dissection, they can be based on different protocol usage (e.g. IPv4 vs. IPv6 addresses), VPN of any kind, the 5 types of NAT, access operator specific networks and realms and also they may represent and establish business models as defined by economic factors such as pricing. Security boundaries play a special role because for this type of separation the reliability of the enforcement of the separation of sets of nodes is the priority.

Means for Structuring

Structures may be of static or dynamic nature. Logical hierarchy is seen as a necessity, but it shall have more flexibility compared to today’s situation, where hierarchies are often static throughout entire lifecycles of networks. A first step has been to identify how the potentially extremely large set of nodes can be separated into smaller sets at all, adhering to the INM paradigm. This is one feature of the self-organizing behaviour of the INM entities, in addition to the others, better known “self-x” features; this is evident for example in Section 5.2.1.

Structures can be seen as relationships between nodes, groups of nodes and boundaries between the groups, as known from the theory of sets. The structures may be established for certain contexts: one may be used to formally separate operators from one another, and others may be temporary groups of nodes that are created to deal with certain load conditions for a limited amount of time, or for troubleshooting. The basic idea of collaborating groups is also described in Section 4.5**.** The main idea is to establish the groups by a topology discovery process and then let the members of the collaboration group detect any internal anomalies.

Considering the different requirements presented above, we can distinguish between two types of initiation: outside-in or inside-out. The outside-in initiation is adopted for structures derived from operator's objective, and can therefore be configured through the GMP point. The inside-out initiation is adopted for management operations coupled with the network functions and it creates structures between cooperating neighbouring nodes which group themselves subsequently into larger conglomerates. The co-existence of the two methods enables scalability and guarantees a high level of flexibility in the aggregation of management operations. See Figure 3-13 for the basic idea.

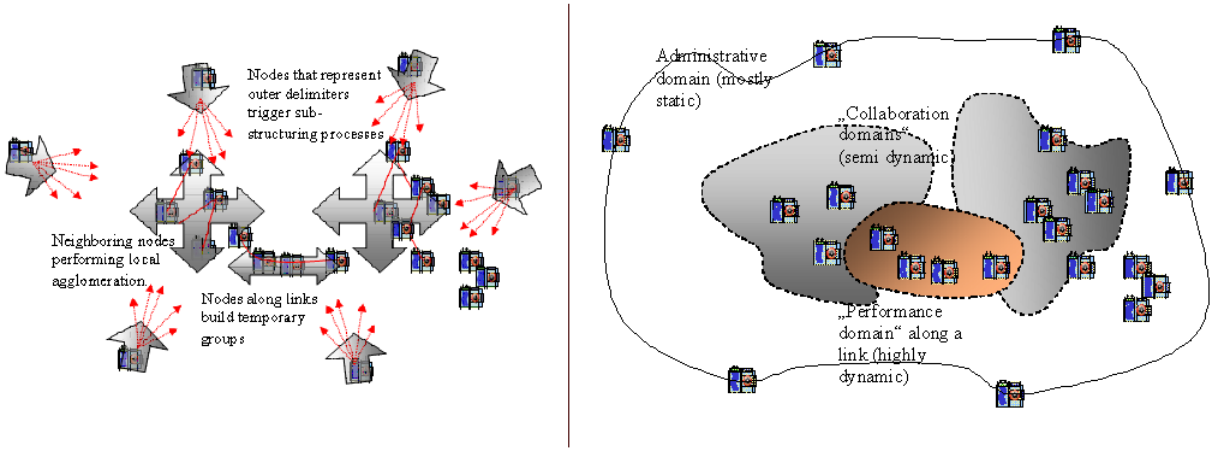


Figure 3-13: Emergence of structure by using outside-in and inside-out self-structuring processes of an INM-enabled network (left: preparation signalling; right: established domains)

Technical means for creating boundaries for NM include:

- Gossip-based algorithms, like those presented in Chapter 4, can be adopted in networks with no configuration retrieved from the operator or services;
- Algorithmic creation of structures without or with limited central control can be performed using self-organizing algorithms;
- Graph-colouring and/or cellular automata taken from computer science fields;
- Algorithms for finding a master node within a set of nodes are very well known (such as letting every node create a random number, and the highest one wins);
- An extension of this is the automatic creation of sub-sets of nodes, according to certain criteria, for example the node with the highest ID selects a pre-defined number of neighbours, and the voting starts again in the remaining set of nodes (not including the first created domain, i.e. the first master and its neighbour nodes). The master role may change more or less frequently over time.

A structure, initiated by one of the two methods described above, can be enforced into the management capabilities at different levels, either as information elements encoded in the network nodes or as soft-state information, like a "scope" for specific management functions. Both implementations have pros and cons, and the INM architecture can adopt both for better performances. From the analysis of the requirements, it is proposed to map the outside-in initiated structures into information elements inside the INM kernel. They can then be used from the management capabilities as constraints to define more dynamic structures, creating temporary scopes for specific management operations.

The INM node architecture should reflect such structures, ensuring that any node's INM kernel and/or INM application space can be made aware of belonging to certain domains, or taking an active role in the implementation or maintenance of a certain boundary between domains. We propose to define a dedicated management FC (dmFC) that takes care of the awareness of any node or SE (ME) for belonging to any groups or larger scale domains, i.e. managing membership and triggering of structuring processes according to certain predefined goals. It has to be analyzed whether the INM registry can also be used to implement such functionality, i.e. awareness of nodes/SE themselves as well as groups, domains and overall structures.

3.6 Information Management and Interworking with NetInf (WP6)

Experience from previous architectures [41] and [42] shows that several aspects related to information handling are important for quantitative aspects, like timeliness and memory consumption, as well as qualitative aspects, like ease to define objectives in the network and usability of management instruments. The analysis work of deliverable D4.1 provides examples of such technical issues. Scenario 1: given the limited computation capabilities of the sensors, which dissemination mechanism performs best for configuration and measurements? Scenario 2: how to abstract and map at different levels the information to control large scale networks? To address the variety of these aspects, we adopt a similar approach as in [43] and we discern the design aspects pertaining to information handling with the design aspects pertaining to functions.

This section addresses network management information storage and retrieval in decentralized settings. We consider decentralization of both the information storage as well as the management functionality itself. Both work in a very centralized and hierarchical way traditionally in large telecommunication systems, in line with the well-known pyramid-structure (defined by the TM Forum [69]), where network elements (NE) are handled by element managers (EM), which are in turn controlled by a network manager (NM), and even higher layers for service (SM) and business management (BM).

In most types of networks, the management information source can be any network element. Therefore, a decentralized storage might adapt better to the decentralized nature of the information, especially if the information is needed locally only. For all other cases, the pros and cons have to be elaborated and benchmarked with centralized designs.

The expected improvement is storage of INM data, information and potentially state in a way that has benefits in reliability, access speed, and query-like search options. We want to benefit from those properties for NM.

The Path to Decentralized, In-Network Management and Related Work

Traditionally, the NM is logically and also location-wise fairly centralized, and, correspondingly, any management related information is retrieved from the network through some management protocols and typically stored in a central database. Also the NM system is in charge of retrieving the information from the network, processing it and storing it in the database. Another characteristic is the hierarchy in traditional NM; every network element (NE) has a relationship with an element manager (EM), which in turn is connected to the overall, topmost NM system. The first set of enhancement had been the decoupling of the NM functionality from the retrieval and setting processes. Most prominent work has been done by J. Strassner [10] in the context of Directory-enabled Networking (DEN-ng). Still, the storage is central, but accessible remotely from any NM application, typically through LDAP or any other protocols.

On the other hand, the way towards decentralized management has been paved through various work on Management by Delegation [12], the IETF Script MIB [13], and patterns for decentralized management [11] increasing the degree of decentralization.

Madeira [9] suggests using the “P2P principles” for distributed NM and follow a model-driven approach. However, they do not provide details on storage procedures. Also the class of end-system based P2P management paradigms have been worked on [15], but they don’t couple this with network wide systems.

The in-network management paradigm can be interpreted as pushing management intelligence into the network, and, as a consequence, making the network more intelligent: as a consequence, objectives and costs of management operations can be adapted according to local working conditions. Glued together with a set of discovery and self-organizing algorithms, the network elements form a thin “management plane” embedded in the network itself.

While it is too early for specification of data types for INM storage, the potential data to be made available to other SE/ME (also related or depending on their domain membership) includes: (i) Measurements, specifically accumulated lower layer measurements of interest to

other nodes; (ii) Measurements / results (aggregations) that have been created by gossip protocols and may be of interest later, or to other nodes (heuristic storage, to avoid new or too frequent measurements); (iii) Data about domains/structures / network data base; (iv) Context information with more or less tight timing constraints; (v) Storage of MCs description (cf. Section 3.5.2 on functional issues; (vi) Service specific data.

3.6.1 Data Storage through NetInf

Here we consider WP6's Network of Information (NetInf) approach to information storage and handling, which are based on separating content and locator information and applying one or multiple indirection steps, resulting in the ability to store and retrieve information objects in a more generic way. NetInf is foreseen to provide a storage middleware allowing to store information elements in a decentralized way and provide search and retrieval capabilities. It can be seen as a generalization of various systems in the P2P, CDN, and file sharing space. Since this particular system is under design at the moment, we focus on the requirements and challenges are for storing and retrieving NM information with it.

Network of Information Background

The overall objective for a NetInf [8] is to design a general, information-centric network architecture for information retrieval and storage. It is concerned with the information objects themselves rather than the nodes that host them. Information objects are directly addressed, without any knowledge of what node they are actually hosted. The main components of a NetInf are a modelling framework that facilitates object discovery, lexical retrieval and syntax and semantics of object operations. The architecture is based on two boundaries, a lower API towards the network infrastructure, such as IP, and an upper level API, which is meant to be used by future applications. In our case, we assume that the management functionality uses the upper API for the storage as well as the retrieval of management information.

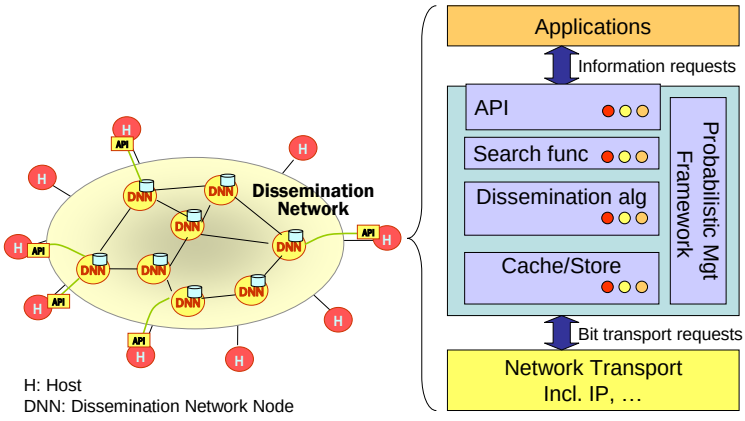


Figure 3-14: Network of Information according to WP6 overview

Most peer-to-peer systems are only used for simple non-wild carded searching of information rather than storing the information itself. Also many do depend on distributed hash tables (DHT) in their core. However, that depends on the specific application it is used for. The NetInf system is foreseen to provide a larger breadth of functionality including decentralized storage, but also search and retrieval functionality. We assume the decentralized storage system being a high-level concept, currently under design. It has the capability to put information into and to retrieve information from the system.

Typically, NetInf systems are not targeted to NM information handling but rather to end user content such as multimedia. Many of the search and retrieval mechanisms in NetInf systems are built for that specific purpose, and adaptations might be needed on both sides, i.e. on INM side as well as on NetInf side, in order to exploit the concept. Adaptations on INM side include re-formulation of NM information flows and procedures, while on NetInf side, certain requirements such as fast access times may be important.

3.6.2 Data Storage through Situation Awareness Framework

The situation awareness framework offers another possibility to store and offer data for INM purposes. The whole framework is described in more detail at Section 4.3. The basic idea of this approach is a decentralized directory that is aware about the location of all offered INM data. Each entity, INM application or any component that offers INM information registers itself to the directory indicating the sort and location of the information offered. Vice-versa, each entity, application or component is able to retrieve needed INM data by sending a resolve to the directory which replies with the correspondent location of the requested information. Using the location information, INM data can be requested directly P2P afterwards, which can be requested either by a get or subscribe. In the former case the INM data will be sent just once, in the latter one it will be sent continuously every time new data is available.

As the name already states the main purpose of the Situation Awareness Framework is offering situation information that helps to establish situation awareness. On the one hand, such information can be either low level information such as BER, link quality or level of congestion. On the other hand information can be also offered as aggregated or interpreted, e.g. anomalies or group decision. This way it would be also possible to offer NetInf information via the situation awareness framework. NetInf data would just need to get registered at the directory to be available to every component that wants to retrieve such data.

3.7 Use Cases for the In-Network Management Framework

The objective of this section is to provide concrete examples of how the INM framework can be applied. Section 3.7.1 shows how two management algorithms, topology discovery and anomaly detection, are placed in the context of the framework. Section 3.7.2 presents several example instantiations of INM, specifically, an example of a QoS functional component.

3.7.1 Implementation Example of Algorithms in the Framework

In this example, we describe the design and implementation considerations of two algorithms, topology discovery and anomaly detection, as well as the collaboration between the two in the context of the framework.

Topology discovery is an essential management capability (MC) for self-management, which is capable of enabling a host of other MCs. An inherent feature of self-management is that management tasks are taken care of locally and autonomously by the network nodes (the self-managing entities - SEs), which also implies a distributed effort.

Every network node is part of one or more local collaboration groups, with which it collectively implement some other MC. Upon bootstrapping, every SE initiates a process that discovers its neighbours, for the purpose of creating a collaboration group, or joining with an existing one. An SE can be part of multiple collaboration groups, implementing different MCs. Members of the collaboration groups can be selected based on certain criteria, such as nodes that provide services to other nodes, or based on the class of the nodes or how far they are from each other with respect to some topological distance. Mechanisms for adding/removing members from the collaboration groups should be supported, as well as discovery, member negotiation, and collaboration with other collaboration groups.

Anomaly detection, besides its importance for autonomous health management in network elements, also can be used for resource management. This means that the detection of anomalous behaviour on a link or a node can trigger self-managing mechanisms, such as re-configuration. For these purposes, an anomaly detection MC can be used by different functional components (FCs) and, if necessary, collaborate with other MCs.

In our example, an anomaly detection MC offers the capability of detecting and reporting faults on link or node level in the network. The algorithm implemented by the MC operates in a distributed manner in individual network elements, and collaborates with other network elements that implement the anomaly detection MC. The anomaly detection algorithm actively sends probes for both monitoring of neighbouring nodes and for classification of detected

anomalies, which is done by requesting assistance from neighbours of the detected anomalous node. For these purposes, the anomaly detection algorithm needs network topology information which can be obtained by topology discovery algorithms. In terms of the INM framework, the anomaly detection MC must be able to communicate with the topology discovery MC in order to form collaboration groups of neighbouring nodes within the topological distance of two.

Design Considerations

Due to the distributed nature of INM, every network element makes use of topology discovery. Since it is a generic MC, an inherent degree of embedding is surely excluded. The topology discovery MC can be operating in two different setups:

1. Every network element has at least one FC that includes the topology discovery MC. Through the discovery and negotiation process, one of the FCs is taking a lead role in forming the collaboration group (i.e. the manager of the collaboration group). For a multitude of reasons, the manager of the collaboration group can be changed (namely, moved to another network element). In this setup, the topology discovery MC is attached to a self-managing FC (smFC) or a dedicated management FC (dmFC), using either the integrated or the separated degree of embedding, respectively.
2. In every network segment, there will be one or more designated network element with an FC that manages the collaboration groups. Those FCs will be the only FCs in every network element in the segment that include the topology discovery MC, most likely, as a dmFC, under the separated degree of embedding. Note that such dmFC can, but not necessarily do, implement other MCs. All other network elements in such a network segment have FCs that are equipped with one or more atomic or primitive MCs that allow them to participate in the group formation process.

Both setups are possible. The first setup is more flexible, as every network element has an FC that can become the manager of the grouping process. However, that means that the topology discovery MC is attached to at least one FC in every network element. The decision which setup to implement is a design issue.

The topology discovery capability can be implemented by a few MCs. The number of MCs is considered a design decision. One could separate the discovery capability from the formation of the collaboration groups, and possibly attach them to different FCs.

The anomaly detection capability can be implemented by three or four MCs. Essentially, network elements that will perform anomaly detection need an FC that implements an anomaly detection MC. The FC that hosts the anomaly detection MC communicates (through the collaboration interface) with members of the collaboration group of network elements formed by the topology discovery MC. Apart from collaborating with the topology discovery MC, for the purpose of exchanging topology information, the anomaly detection MC needs (for monitoring purposes) to communicate with another FC and/or MC that can provide indicative information that relates to the health of other members in the group (in this case used for probing of nodes). This communication can be done either through the collaboration interface of an FC that contains an MC that implements probing functionality, or directly with an MC through accessible probing functions. In addition, the anomaly detection MC needs to implement functions used for notifying collaborating network elements about the status of neighbouring network elements, and for confirmation of failures on request. Such methods are made accessible via the collaboration interface of the FC that hosts the anomaly detection MC. Further, configuration parameters of the anomaly detection MC can be set via the organization interface of the FC. The anomaly detection MC also requires means to report faults. This can be done by making report functions implemented either in the anomaly detection MC itself or in another MC that the anomaly detection MC collaborates with. Depending on the design, the anomaly detection MC could be included under the integrated degree of embedding in an smFC, in order to make the implemented report function public and accessible via the service interface (provided by the smFC). On the other hand, the

anomaly detection MC could also be included in a dmFC under the separated degree of embedding. In that case, the anomaly detection MC communicates through the collaboration interface (provided by the dmFC) with other FCs (smFCs) that provide interfaces for user services, and that contain an MC that implement reporting functionality.

The FCs and MCs described can either implement a single SE used for the purpose of anomaly detection only, or be included as part of an existing SE designed for e.g. in-network monitoring. The approach to take is a design choice made by the developer of the SE.

Implementation Example

In the implementation example that we show below, both topology discovery and anomaly detection is included in both types of FCs (i.e. smFCs and dmFCs). Further, the topology discovery MC is included in the FCs following the first setup described above. Figure 3-15 demonstrates the described arrangement of this example. In the figure, each network element within the collaboration group is represented only by the FC that implements anomaly detection (AD) and topology discovery (TD). The TD MC (shown as the larger bubbles on the right inside each one of the FCs) communicate with other nodes in the area, negotiating a TD manager, and eventually forming an AD collaboration group. Such communication is shown by the lower lines. The AD MC (the larger bubbles on the left included in the same FC), communicates with other members of the AD collaboration group (shown by the upper lines).

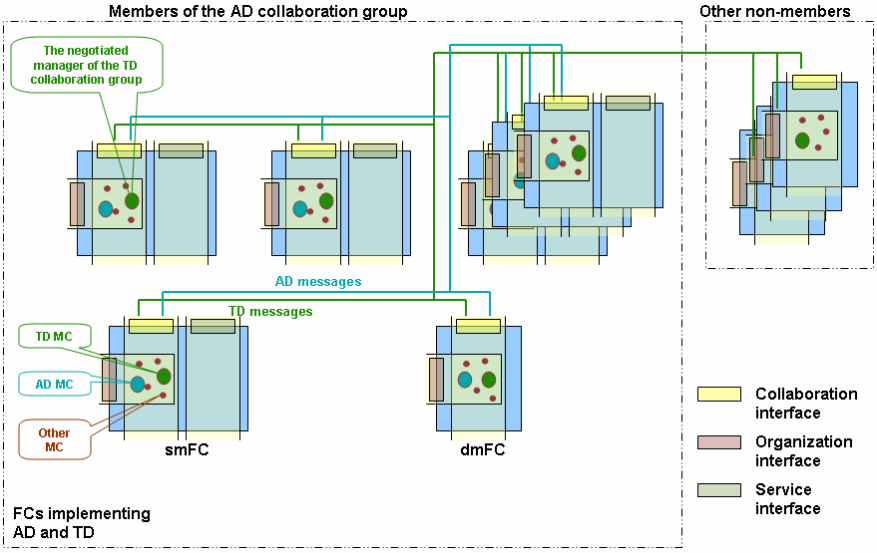


Figure 3-15: Example of topology discovery and anomaly detection in the framework

3.7.2 Example of INM Framework Instantiation

The framework instantiations presented in this section are the first feasibility study to verify the adoption of the INM framework in practical use cases. They are based on Figure 3-16, showing the instantiation of the framework into management capabilities distributed across different nodes.

Interactions with Service Components

Instantiations of the framework are discussed with an example related to a dedicated management functional component (dmFC), such as quality of service (QoS). We use the following distinction to emphasize the role of dmFCs:

- Self-managing functional components (smFCs): NetInf, GP, Vnet;
- Mandatory dmFCs (for any node) : QoS module, routing module;
- Optional dmFCs (for strategic nodes only): resource control module, security module, neighbour discovery module, etc. Note that a strategic node is usually located in a mediation point and it includes more dmFCs than the mandatory QoS and routing.

These functional components communicate through several interfaces: smFCs through service interfaces, whilst dmFCs use the collaboration and organization interfaces. However there is also an interaction between smFCs and dmFCs inside the INM kernel, but also inside INM applications. Users may have access to smFCs and providers could interact with dmFCs (SLA-based). There are GPs between nodes and dedicated GPs for inter-management.

The management capabilities included in the functional components (driven by one or several processors, depending on INM kernel implementation) are listed in Table 3-2.

Management capabilities	Interface	Beneficiaries
Physical resources access	service	framework to perform inter-process communication (IPC). INM will permit/ deny access to hardware.
Physical resources access	collaboration	QoS module to perform(technology dependent) measurements between nodes within the defined scope (intra-domain, inter-domain) through hardware.
Cross-layering for committed QoS parameters	collaboration	QoS module to impose a transfer rate to the hardware through INM platform and hardware driver
Cross-layering for QoS parameters	collaboration	QoS module to collect BER, BER time distribution, nominal transfer rate, etc. from hardware driver through INM platform
Cross-layering for data	service	Self-managing functional components may exchange information to/from hardware driver through INM platform. This is crucial in case of emergency.
Real-time composite metric calculation	collaboration	QoS module to provide the composite metric for other INM modules only (routing, resource control, security, etc.)
Real-time composite metric calculation	service	QoS module to provide the QoS service requested by framework (self-managing FC like NetInf, Vnet, GP)
Routing for management	collaboration	Routing module to offer services to resource control module (optimal routing, policies); neighbour discovery module; security module
Routing for management	collaboration	other modules from other nodes; ForMux to establish dedicated GPs for management (BER less than a threshold, best coding schemes, swarm-like with multicast services, multipoint-to-multipoint)
Routing at node level	service	framework (dmFCs like NetInf, Vnet, GP) to perform virtual routing
Routing at node level	service	framework in case of emergency (DEFCON [1])
Resource planning	service	ForMux to perform GP management; Vnet to perform Vnet management; NetInf to perform NetInf management
Resource Planning	service	infrastructure providers (via Vnet), operators (via Vnet)

Table 3-2 Examples of management capabilities

Example of Dedicated Management FC: QoS

The network of the future is believed to be a network with converged services. The same network will provide access to data, voice or high quality video content to the end users. Because not all the data flows require the same traffic parameters, we have to classify them according to the transported information type and treat them differently. All these operations are made in order to maintain a specific quality of the service, provided by the operator to the end user and specified in the SLA between those two entities. Because a usual traffic flow crosses different communication domains, each one with its own rules, it is still a challenge to guarantee end-to-end QoS services on a communication channel, because each domain can

implement different QoS mechanisms that are not always compatible, or worse, are not offering QoS at all. The reason that many network administrators choose not to enable QoS functions in their network is that they are difficult to configure properly, requiring a thorough understanding of the mechanisms behind. We consider that QoS is an important management functionality that should be supported by the framework developed in 4WARD.

The architecture of each network node should contain a quality of service dedicated management FC (QoS dmFC), that handles all functions related to guarantying a specific quality of services, offered to the users. The QoS dmFC will have 2 interfaces: service and collaboration, and it will implement three groups of management capabilities (cf. Table 3-2): *accessing the physical resources*, *cross-layering* for QoS parameters and/or data, *composite metric calculation*. The first MC implemented by this module will enable QoS dmFCs to perform measurements between nodes within the defined scope (intra-domain, inter-domain), accessing the hardware directly, through the collaboration interface. The idea of second MC, *cross-layering,* refers to the optimization of a protocol in one layer, due to changes in the quality of the service of other layers. This capability will be implemented using 2 approaches:

- **Bottom-up approach:** will enable collecting traffic parameters like: BER, BER time distribution, nominal transfer rate that are characterizing a specific physical link and are an objective way of evaluating a communication channel. The results could be obtained directly from the hardware driver where the technology will permit, or using different dedicated tools that will perform passive or active measurements between nodes. The information regarding the performed measurements will be exchanged between the QoS dmFC and the hardware through the collaboration interface.

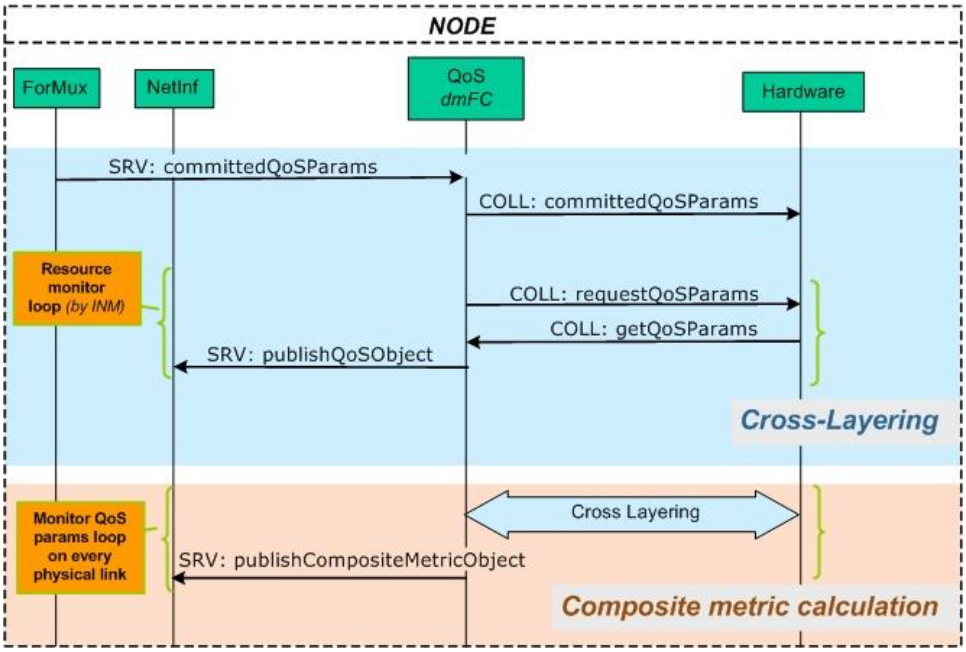


Figure 3-16: Example of exchanged messages between functional components (cf. Annex A)

- **Top-down approach:** will impose a specific transfer rate to the hardware through INM platform and hardware driver. The traffic parameters will be received from different applications (e.g. GP) through the service interface and send directly to the hardware, using the collaboration interface.

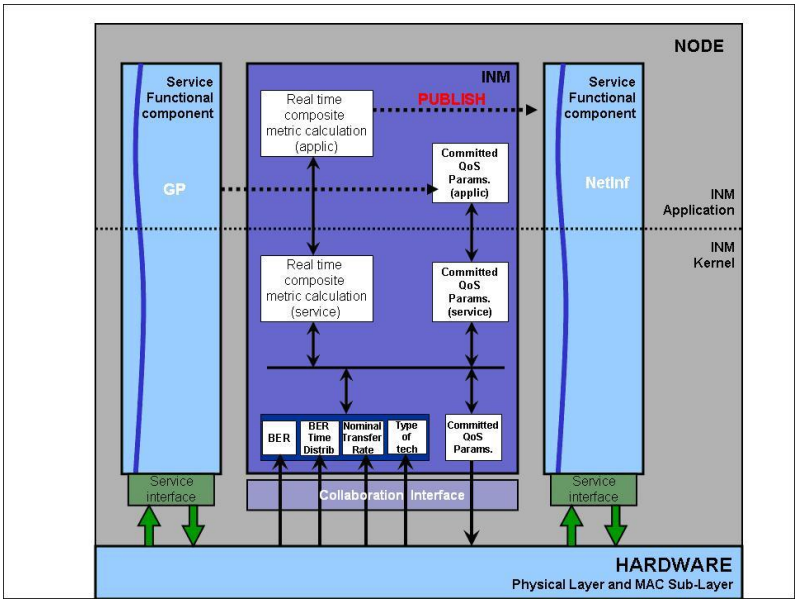


Figure 3-17: Node instantiations: details regarding QoS module

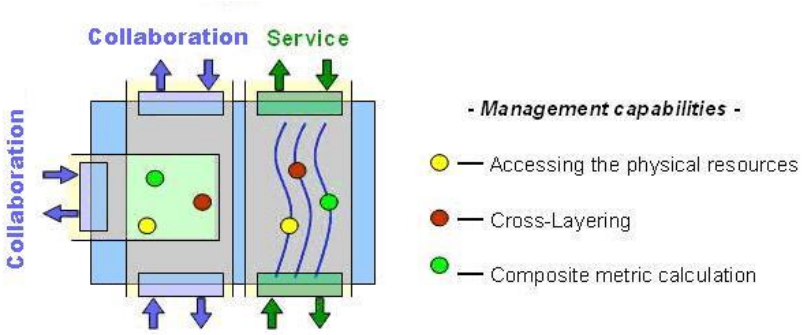


Figure 3-18: QoS dmFC

The third management capability in the QoS module is *composite metric calculation*. The metric will be calculated using the traffic parameters, received from hardware or measured using dedicated sub-modules and provided as a service to the routing module (dmFC) or other service functional components like NetInf, Vnet or ForMux if requested. As a final step, the global information obtained for management purposes (not for operational tasks) will be published into NetInf through the service interface, so that every network entity to be able to acquire the metric that characterizes a certain physical link. Because the performance of a communication channel varies in time, the QoS dmFC will constantly perform measurements and recalculate the composite metric, updating its value.

Considering that the future network should be able to manage itself with minimum intervention from an administrator or a governor, while maintaining good performance, the QoS dmFC should be characterized by the next set of properties: self-descriptive, self-configuration, self-optimization, composability, governance, interoperability and extendibility.

4 Distributed Algorithms for Real-time Monitoring, Anomaly Detection and Situation Awareness

Monitoring, anomaly detection, and the creation of situation awareness are crucial functions of the INM management plane, as they support other management tasks, including fault, configuration, accounting, performance, and security management. Since the INM paradigm aims at providing effective management for large-scale, dynamic network environments, these functions must operate in (near) real-time.

In a dynamic environment, a network must continuously adapt its configuration, in order to maintain its state near a desired operating point, despite perturbations caused by load changes, failures, etc. To achieve this, a set of algorithms is needed for estimating the network state with the needed accuracy. Specifically, the required functionality includes:

1. Monitoring of network-wide aggregates in real-time. Aggregates contain information about the network state, such as the current number of VoIP flows in a network domain or the current bandwidth consumption by different types of applications.
2. Probabilistic anomaly detection. In an autonomous setting, it is important to devise distributed and localized approaches to fault detection and localization.
3. Real-time resource and traffic monitoring in support of routing, specifically resource discovery, estimation of available resources (quantity and quality) and resource consumption.
4. Situation awareness, e.g., the process of extracting and evaluating raw sensor data and monitored aggregates, to drive to the decision making processes in the management plane.

The algorithms provide the necessary input to the management plane to perform self-adaptation tasks that are further discussed in Chapter 5, as well as to control algorithms developed in other WP's of the project. For instance, the algorithms for real-time monitoring for routing (Section 4.2) can provide input to WP5 for the configuration of generic paths.

The present chapter reports on progress in the above four areas. Specifically:

- In the area of distributed monitoring, several algorithms have been developed that examine different types of aggregates including threshold crossing indicators and histograms. These algorithms have been evaluated and compared using analytical and experimental techniques to assess performance, overhead and robustness.
- We have developed a method for collaborative and distributive identification of network anomalies, as well as faulty nodes and links, using end-to-end test transactions or probes. As we cannot rely on manual configuration of parameters, the algorithm autonomously adapts to network properties.
- We report on a preliminary study on resource and traffic monitoring for routing in the Future Internet.
- We outline a model for situation awareness that allows algorithms for distributed monitoring and anomaly detection to be embedded.

4.1 Tree-based vs. Gossip-based Algorithms for Monitoring Aggregates

Monitoring, i.e., the process of acquiring state information from a network or networked system, is fundamental to system operation. In traditional network and systems management, monitoring is performed on a per-device basis, whereby a monitoring station periodically polls devices for the values of local variables, such as device counters or performance metrics. These variables are then processed on the management station to compute an estimate of a network-wide state, which is analyzed and acted upon by other management programs. SNMP is probably the best-known protocol that supports this monitoring paradigm.

Over the past 20 years, this paradigm has proved fairly successful for networks of moderate size, whose configuration rarely change and whose states evolve slowly and thus do not

require intervention within seconds by an outside system. These assumptions, however, do not hold for the networks of today, and neither will they hold for the Future Internet. In the following, we outline our thoughts on a monitoring system for networks that are very large, whose configuration changes frequently, and whose state is highly dynamic and thus must be available at control points with short delay.

To ensure scalability and fast reaction times, the processing associated with monitoring should be carried out inside the network. We thus advocate research towards a light-weight, distributed management layer inside the network that offers end-to-end monitoring primitives to management applications and end systems outside the network.

Aggregates contain information about the state of an entire system, as opposed to that of a single device, and many management applications depend on such data. For the purpose of quality assurance, for instance, it may be required to continuously track the number of VoIP flows in a network domain or the distribution of traffic composition across all links. Similarly, to achieve a given level of availability, it may be necessary to know, at all times, the percentage of links that operate above 50% utilization and to identify the 10 most loaded links.

The best-known approach to computing aggregates in a distributed fashion involves creating and maintaining a spanning tree and aggregating state information along that tree, bottom-up from the leaves towards the root. Such a tree can be built in a decentralized, self-stabilizing manner, which guarantees robustness in the monitoring protocol. A second, less-studied approach involves the use of gossip protocols, which typically rely on randomized communication to disseminate and process state information in a network.

While both types of protocols execute on a network graph, which can be realized as an overlay, there are significant differences between tree-based and gossip-based aggregation. First, gossip-based aggregation protocols tend to be simpler as they do not maintain a distributed tree. Second, tree-based protocols generally deliver the result of an aggregation operation at the root node. Gossip-based aggregation, on the other hand, produce estimates of the aggregate at all nodes. Third, node failures are handled very differently for the two protocol types. Tree-based protocols are often designed to be robust against node failure, and obtain this robustness by dynamically reconfiguring the aggregation tree. For gossip protocols, node failures can cause information loss, which causes a bias in the aggregation process and needs to be corrected.

An important issue is therefore to examine the suitability of the two approaches for aggregation tasks arising in large dynamic networks, and to compare their performance regarding metrics such as accuracy and adaptability. Contributions have been made in the following areas:

- Gossip-based protocols that are robust against node failures (Section 4.1.1)
- Comparing tree-based and gossip-based aggregation in the context of MANET's (Section 4.1.2)
- Gossip-based solutions to threshold detection (Section 4.1.3)
- Tree-based aggregation of histograms (Section 4.1.4)
- Tree-based aggregation with controllable accuracy (Section 4.1.5)

In the remainder of this section we summarize the results obtained in these areas.

4.1.1 Robust Gossiping for Network Management

Gossip protocols, also known as epidemic protocols, can be characterized by asynchronous and often randomized communication among nodes in a network [44][45]. Originally, they have been proposed for disseminating information in large dynamic environments, and, more recently, they have been applied to various tasks, including constructing robust overlays [46] and estimating the network size [47].

A gossip protocol for monitoring network-wide aggregates executes in the context of a decentralized management architecture. In this architecture, monitoring nodes with identical

functionality organize themselves into a management overlay. The aggregation protocol (in this case, the gossip protocol) runs in the monitoring nodes, which communicate via the overlay. Each monitoring node collects data from one or more network devices. The protocol aggregates this data, in a decentralized fashion, to estimate the SUM, MAX, AVERAGE, etc., of the device variables. A management station or an application server can access the management overlay at any node. Node or link failures—on the physical network or the management overlay—cause recovery actions in the management overlay, thereby enabling continuous operation. This allows a gossip-based management overlay to handle not only fixed networks where failures can generally be assumed to be infrequent, but also networks such as MANETs with a high degree of node and link dynamicity, in this way covering a wide set of scenarios related to the Future Internet.

We have designed a protocol, G-GAP, which hardens the Push-Synopses protocol due to Kempe et al. [44] by making it robust to certain classes of node failures. In Push-Synopses, each node *i* has two local state variables s_i and w_i. At initialization time, when computing the aggregation function AVERAGE, s_i is set to the value of the local variable being aggregated and w_i is set to 1. Then the node sends the pairs to itself to start the computation.

Round 0 {
1. $s_i = x_i$;
2. $w_i = 1$;
3. send (s_i, w_i) to self }

Round $r+1$ {
1. Let $\{(s_i^*, w_i^*)\}$ be all pairs sent to i during round r
2. $s_i = \sum_l s_l^*$; $w_i = \sum_l w_l^*$
3. choose shares $\alpha_{i,j} \geq 0$ for all nodes j such that $\sum_j \alpha_{i,j} = 1$
4. for all j send $(\alpha_{i,j} * s_i, \alpha_{i,j} * w_i)$ to each j }

Figure 4-1: Push-Synopses, pseudo-code for node i

The protocol is a round-based protocol where nodes execute the protocol shown in Figure 4-1. The estimate of the aggregate on a node *i* for a given round is computed as s_i/w_i and this ratio is shown to converge exponentially fast for the case of uniform gossip. The correctness of the protocol (in the sense that the estimate converges to the true value of the aggregate) relies crucially on invariants which express “mass conservation”. These invariants state that, for all rounds, the sum of the variables s_i and w_i, respectively, remain constant or, more precisely, that $\sum_i s_i = \sum_i x_i$ and $\sum_i w_i = N$ (for N the number of nodes in the network).

The Push-Synopses protocol has two major limitations: the protocol does not support continuous monitoring of aggregates (i.e., if the local value changes, this change is not reflected in the computed aggregate) and the protocol is not robust to node failures. These limitations are basically results of violations of the above invariants, which would result in “mass-loss”.

The G-GAP protocol extends Push-Synopses in two directions. First, we adapt “Push-Synopses” to continuous monitoring. Second, and more importantly, we extend the protocol with a scheme to provide accurate estimates in the event of (many classes of) node failures. Due to space constraints we refer the reader to [48] for the details of these extensions, the analysis of the convergence and proof of correctness of the protocol.

We have evaluated G-GAP through extensive simulations using the SIMPSON simulator [51], a discrete event simulator that allows us to simulate packet exchanges over large network topologies and packet processing on the network nodes. In various scenarios, we measure the average relative estimation error by G-GAP on the network nodes, in function of the round rate, the network size, and the failure rate, in order to evaluate the protocol against our design objectives. (Note that estimation errors may incur costs. Examples of such costs include lost

revenue due to under utilization or penalties because of violated SLAs.) In addition to G-GAP, we run most simulation scenarios also with GAP [49], a tree-based protocol we have examined in work prior to 4WARD. This allows us to compare the use of a gossip protocol with a protocol that is based on spanning trees for the purpose of monitoring network-wide aggregates. To make the comparison fair, we measure the performance metrics of both protocols for a comparable overhead. We present here results from two scenarios. The complete evaluation is reported in [48].

In the first scenario, we measure the average estimation error and the 90th percentile of G-GAP and GAP in function of the network size. We run simulations with network graphs generated by GoCast [46], a gossip-based protocol for creation and maintenance of overlay networks with fixed connectivity. We have run experiments for networks of size 82,164,327,654, 1308, 2626 and 5232 nodes with a GoCast target connectivity of 10. The monitored variable represents the current average number of HTTP flows in the network. We simulate the behaviour of the local variables based on packet traces captured at the University of Twente [52]. The results are shown in Figure 4-2. We observe that for both protocols, the estimation error seems to be independent on the network size. In the general case, for synthetic traces generated by the same (random) process, we would expect such a result for both GAP and G-GAP.

In the second scenario, we evaluate the robustness properties of G-GAP. For this scenario, we use the same local traces as in the scenario above (for the 654-node network described above). For each simulation run, we vary the failure rate from 0 to 10 node failures/sec and measure the average estimation error and its 90th percentile. Failure arrivals are generated by a Poisson process, and failures are uniformly distributed over all running nodes. A node that failed recovers after 10 sec and reappears in the place it had in the overlay before the failure. The results are shown in Figure 4-3. As can be seen from the figure, the estimation error for both GAP and G-GAP increases (almost linearly) with the failure rate. We also see that the slope is steeper and the spread is wider for G-GAP than for GAP.

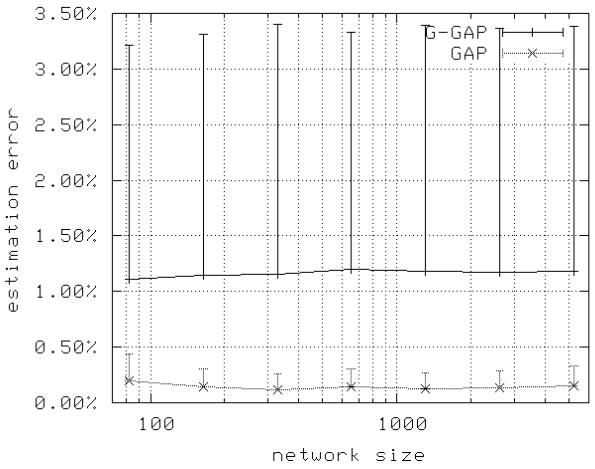


Figure 4-2: Estimation error vs. failure rate for GAP and G-GAP

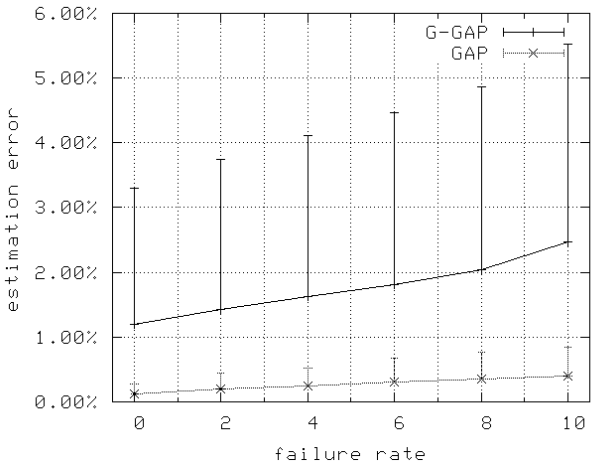


Figure 4-3: Estimation error vs. failure rate for GAP and G-GAP

The simulation results from this work suggest that we have achieved the design goals regarding robustness, overhead, and accuracy. We show that the trade-off between estimation accuracy and protocol overhead can be controlled by varying the protocol round rate. For the traces we used, an estimation error of some 5% or less can be achieved for all network sizes and failure scenarios we simulated. We also observe that the estimation accuracy of the protocol, for a given overhead, does not seem depend on the network size, which makes the protocol scalable. Finally, we prove and validate that the protocol is robust to discontiguous failures.

A second contribution is a comparative assessment of G-GAP with GAP, a fairly standard tree-based aggregation protocol. Our simulation results show that, within the parameter ranges of the simulation scenarios, the tree-based protocol consistently outperforms the gossip-based protocol. For comparable overhead, the tree-based protocol shows a smaller average estimation error and a smaller variance of the error than the gossip-based protocol, independent of network size and independent of frequency of failures that occur in the network.

4.1.2 Gossip-based vs. Tree-based Aggregation for MANETs

In the previous section, we performed a comparison of a tree-based vs. a gossip-based protocol in a fixed network setting. For all scenarios investigated, the tree-based protocol consistently outperformed the gossip-based protocol in terms of accuracy, scalability and robustness.

In this section, we report on a preliminary comparison of the performance of a tree-based and a gossip-based algorithm for continuous estimation of network-wide aggregates in a MANET environment [53]. The performance evaluation is done through simulation, using the NS-2 simulator. In various scenarios, we measure the estimation error by the protocols in function of the protocol overhead, the node mobility, and the number of mobile nodes, for the aggregation function SUM. The protocols are implemented on top of the LLC/MAC layer and hence use the physical layer connectivity as the overlay graph.

As we found out during the simulation runs, for most parameter settings, the version of G-GAP we use performs very poorly due to message losses (between 1-5% depending on the scenario) that is inherent to wireless environments. We decided therefore to evaluate G-GAP under the assumption that no message loss occurs. We call this idealized protocol G-GAP-NL. The performance of G-GAP-NL represents the optimal performance of a protocol that is based on G-GAP.

The performance of the protocols GAP and G-GAP-NL has been evaluated in various scenarios with regards to efficiency, mobility and scalability. The primary evaluation metric used is the average relative estimation error. (As mentioned earlier, estimation errors may incur costs.) The qualitative results of the evaluation are presented in Figure 4-4. We refer the reader to [53] for the details of the simulation setup and the results of the evaluation.

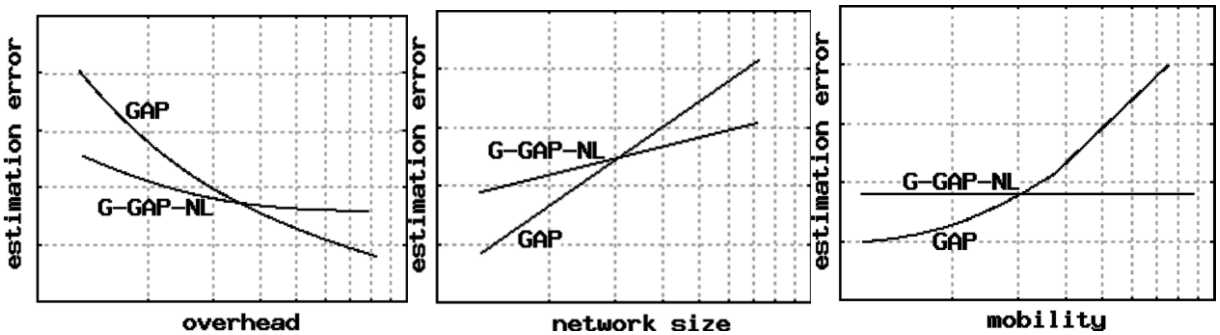


Figure 4-4: Comparison of tree-based (GAP) and gossip-based (G-GAP-NL) protocols for real-time monitoring: qualitative results from simulation study

Regarding efficiency, our studies suggest that at low protocol overhead, the gossip-based protocol gives better estimation accuracy than the tree-based protocol for a given mobility pattern and network size. However, for higher protocol overhead, the tree-based protocol has better estimation accuracy. Regarding mobility, for a given protocol overhead and network size, the tree-based protocol has better estimation accuracy at low mobility while the gossip-based protocol performs better at high mobility. Regarding scalability, for a given protocol overhead and mobility pattern, the tree-based protocol has better estimation accuracy for small networks while the gossip-based protocol performs better for large networks. From the results, we conclude that the gossip-based protocol is better suited for resource-constrained

environments with high mobility and large size. The tree-based protocol is better suited for environments that allow higher protocol overhead, exhibit low mobility and are small in size.

There are two important caveats to these conclusions, however. First, the presence of node failures affects the performance of the G-GAP protocol very significantly, as we have pointed out. Second, more work is required to determine to which extent the results are artefacts of specific implementation decisions, or whether they indeed reveal inherent performance differences between the tree-based and gossip-based approaches. Further work is needed to throw light on this.

4.1.3 Threshold Detection Using Gossiping

Threshold crossing alerts (TCAs) indicate to a management system that a monitored management variable, for instance a MIB object, has crossed a preconfigured value—the threshold. Variables that are monitored for TCAs typically contain performance-related data, such as link utilization or packet drop rates.

In order to avoid repeated TCAs in case the monitored variable oscillates, a threshold T^{g+} is typically accompanied by a second threshold T^{g-} called the hysteresis threshold, set to a lower value. The hysteresis threshold must be crossed, in order to clear the TCA and allow a new TCA to be triggered when the threshold is crossed again (Figure 4-5).

The straightforward approach for the detection of threshold crossings of network-wide aggregates is to use an aggregation protocol to continuously compute the aggregate on a node and to evaluate on that node the threshold conditions every time the aggregate is updated. Several results, both centralized and decentralized, that improve on this approach have been published recently. The common goal is to achieve efficiency by reducing protocol overhead when the aggregate is far from the threshold.

The key idea in adapting a gossip based aggregation protocol to threshold detection is to dynamically adjust the message rate of a node (the rate at which a node communicates updates of its local state) according to the distance of its local estimate of the aggregate to the current threshold. To develop this idea, three largely orthogonal problems need to be resolved. First, the underlying mechanism for dynamic rate adjustment needs to be identified. We propose and explore two mechanisms: reducing the message rate and completely suppressing messages. Second, the mechanism for raising and clearing TCAs needs to be identified. Triggering TCA's on the basis of the local aggregates only is likely to result in an unacceptable level of false positives and false negatives. We propose and explore the use of filters and global snapshot algorithms for this purpose. Finally, the mechanism for exploiting the duality of the problem (i.e., the detection of upward of the upper threshold is technically the same as detecting the downward crossing of the lower hysteresis threshold.) needs to be identified. We propose and explore one such mechanism.

Altogether we identify a design space of 18 protocols, ranging from a baseline design called NNM to the most complex design, called SGM. The NNM protocol uses continuous monitoring without any filtering, and the SGM protocol uses both message filtering/suppression and global snapshots. In [55], the design space is explored, specifically concentrating on three protocols in this space which we have found most interesting. Readers are encouraged to consult [55] for details of the design space and the selected protocols.

We have evaluated key points in the design space outlined above through simulation using SIMPSON [51] through many scenarios where we evaluate the efficiency, the quality of threshold detection, scalability with respect to the number of nodes and the controllability of the protocols. Here we describe one result that demonstrates the efficiency of the protocol.

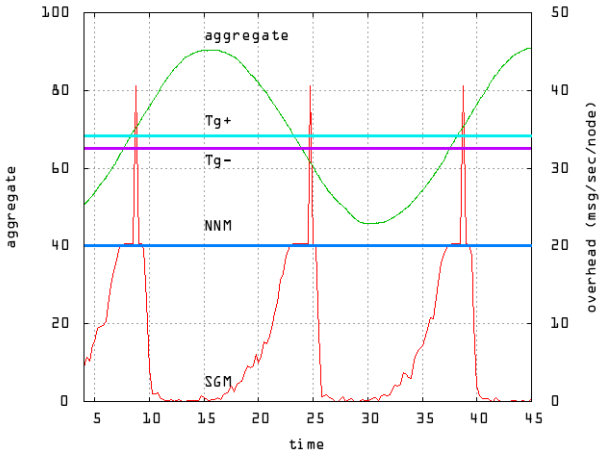


Figure 4-6: The protocol overhead over time for the SGM and NNM protocols

In one scenario, we assess the efficiency of the most advanced protocol SGM against the baseline protocol NNM by measuring the protocol overhead (as the average number of messages processed/sec/node) in a scenario where several threshold crossings occur. We run the protocols on a 654-node network graph generated by GoCast [46]. The simulation is run for 45 seconds and Figure 4-6 shows the trace of the simulation. The figure shows how the monitored aggregate and the protocol overhead vary over time for the two protocols.

Figure 4-6 shows that during the simulation run, three threshold crossings occur: at around t=8.3 sec (upper threshold crossing), t=24 sec (lower threshold crossing) and t=38.2 sec (upper threshold crossing). For the baseline NNM protocol, since no message throttling is employed, the protocol overhead is constant (at around 20 msg/sec/node). For the SGM protocol, before each threshold crossing, e.g., between t=7 sec to t=10 sec, we observe a peak in protocol overhead, as the number of nodes sending messages increases.

The results of the evaluation, at least for the choice of aggregation function and local variables in our simulations, are promising: when the aggregate is far from the threshold the protocol overhead is negligible, and when the aggregate is close to the threshold the overhead is comparable with that of the underlying aggregation protocol. We obtained small detection delays and, for the scenarios considered in this paper, absence of false positives and false negatives. Regarding scalability, at least for the scenarios considered in this paper, we did not observe any significant dependence of detection delay on system size.

4.1.4 Decentralized Real-Time Monitoring of Network-wide Histogram

Distributed solutions for the aggregation of local variables usually push the aggregation function inside the network and aggregate the partial result in a decentralized way. Such solutions prove to be very efficient and exhibit good load balancing properties, along with increased robustness and resilience to network dynamic [59][56][57]. However, they are individually tailored for the specific envisioned functionality and cannot be used in case the management station switches the objective function. E.g., a single instantiation of a distributed protocol designed for aggregating SUM cannot provide an answer for a MAX or MIN query at the management station; hence a new instantiation of a different protocol is required. To address this, one option is to design a distributed protocol that provides the management station with an accurate estimate of the histogram of the monitored local variable. Based on the estimate obtained through a single protocol instantiation, the management station has the

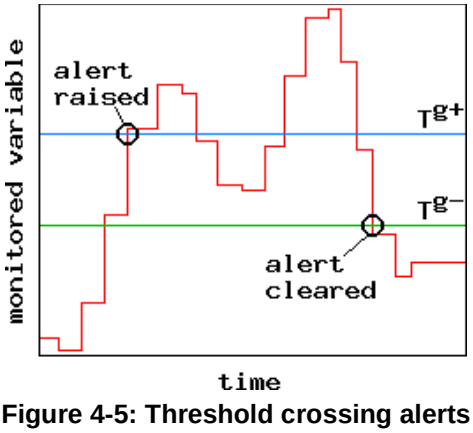


Figure 4-5: Threshold crossing alerts

flexibility of applying locally any desired statistical function, either sequentially, or in parallel, e.g., SUM, MIN and MAX queries can be answered in the same time only through local manipulation of the aggregate histogram.

We address the problem of continuous monitoring of a local variable's histogram with accuracy objectives for large-scale network environments. Our goal is to design an efficient aggregation protocol that allows us to control the trade-off between the accuracy of the estimation and the protocol overhead. Our protocol continuously computes the distribution of the values of a monitored local variable through histogram aggregation by (i) creating and maintaining a self-stabilizing spanning tree and (ii) incrementally aggregating the variable's histogram along the tree (Figure 4-7). It is push-based in the sense that changes in the partial distribution of the monitored variable values are sent towards the management station along the aggregation tree. The protocol controls the management overhead by filtering updates that are sent from nodes to the management station, by allowing for local errors between the last sent updates and the present network status.

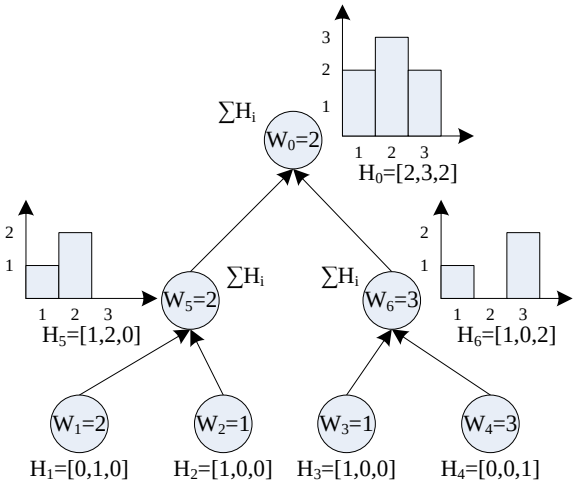


Figure 4-7: Histogram aggregation on trees

The local filters periodically adapt to the dynamics of the monitored variables and the network environment, keeping in mind a global error objective for the monitoring process. All operations in our protocol, including computing the partial histogram of the variable and the configuration of the local error shares, are executed in a decentralized and asynchronous fashion to ensure robustness and achieve scalability. While designing our protocol we keep in mind the goals of (i) controllable accuracy, as a trade-off between aggregation accuracy and protocol overhead, (ii) dynamic adaptation to changes in the network topology, or changes in the evolution of local variables, (iii) controllability in terms of real-time changing the accuracy objective, or protocol response to system changes, and (iv) scalability, as we design our protocol for large distributed systems.

Our proposed heuristic algorithm for error-share allocation comprises an initialization phase, in which the error-shares are distributed to parent nodes in the management tree starting from the root node. Furthermore, it adapts the error-share allocation at runtime, based on monitored changes in the local variable dynamics, and in the network topology [58]. Figure 4-8 presents the protocol overhead results obtained by our algorithm in terms of reduction in maximum link utilization for network graphs of different sizes. We observe that even a small tolerated error in the aggregated histogram leads to a dramatic reduction in the maximum link utilization, hence a significant reduction in protocol overhead. Our solution can control its operation point on the trade-off curve between protocol overhead and system accuracy objective.

4.1.5 Controlling Performance Trade-offs in Adaptive Network Monitoring

A key requirement for autonomic management systems is a short adaptation time to changes in the networking conditions. In a dynamic environment, where networking conditions change fast, a slow-adapting management system lags behind the system state. In such scenarios, a slow-adapting management system provides outdated configurations, thus never reaching a configuration that permits to meet the manager's goals.

In this section, we focus on the adaptation time of a monitoring system to changes in the performance objectives, network topology or networking conditions. We investigate this topic using (as an example) A-GAP, a tree-based distributed protocol for continuous monitoring of global metrics [70]. Based on a stochastic model, the A-GAP protocol dynamically configures local filters that control whether an update is sent towards the root of the tree. These filters are the mechanism we use for trading accuracy and overhead.

We make two main contributions. First, we show that the adaptation time of a distributed monitoring protocol can be controlled. This result adds to our previous work, showing that the performance of a distributed monitoring protocol can be controlled along different performance dimensions, including protocol overhead, protocol accuracy, and adaptation time (Figure 4-9). Second, we demonstrate (through simulation experiments) that there is a trade-off between the adaptation time of the protocol and the protocol overhead in steady-state. This trade-off is controlled by the topology of the tree for the case of A-GAP and we show that the choice of the topology can be used to achieve performance objectives. Generally, allowing a larger protocol overhead permits reducing the adaptation time.

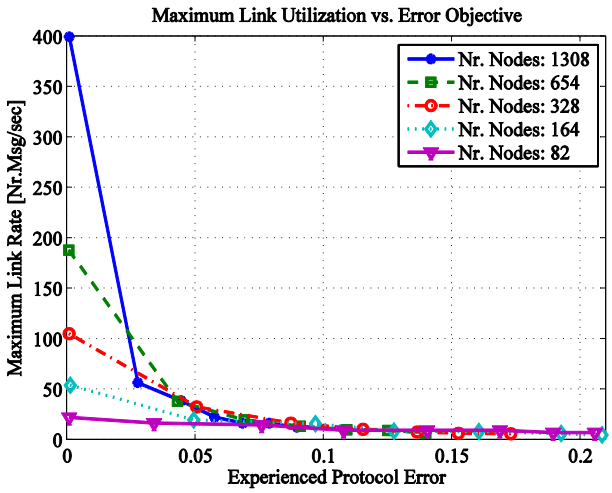


Figure 4-8: Maximum link utilization

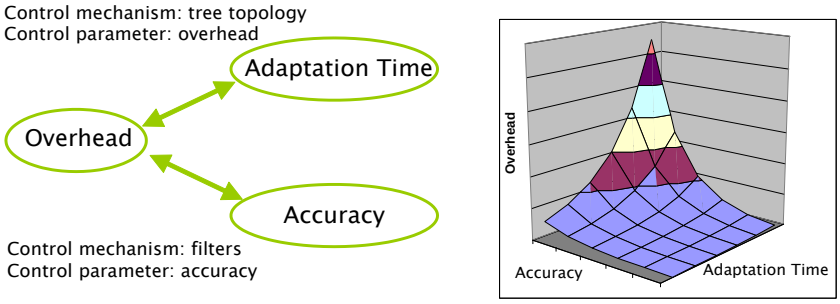


Figure 4-9: Performance trade-offs in decentralized monitoring. The figure on the right shows the border of the feasible region for a protocol such as A-GAP

While it is well known that the topology of the tree strongly influences the performance of monitoring protocols [70] there is no literature on how to effectively control the protocol performance through the tree topology. We have analyzed the performance of A-GAP through extensive simulations using the SIMPSON simulator [51]. Detailed results are reported in [70] which we refer to for detailed information on the simulation setup. Here we bring out the main points only.

We have measured the protocol adaptation time in function of the experienced overhead (the maximum overhead across all nodes). Figure 4-10 shows the measurement results for a network of 200 nodes. Every point in the figure corresponds to a simulation run.

As can be seen, the adaptation time decreases monotonically, as the overhead increases. For smaller overheads, the adaptation time decreases faster than for larger overheads. Consequently, the adaptation time can be reduced by allowing a larger overhead. For example, compared to an overhead of 18,3 updates/sec, allowing an overhead of 20,5 updates/sec (a 10% increase) reduces the adaptation time by 55%. An overhead of 29,1 updates/sec reduces the adaptation time by 97%.

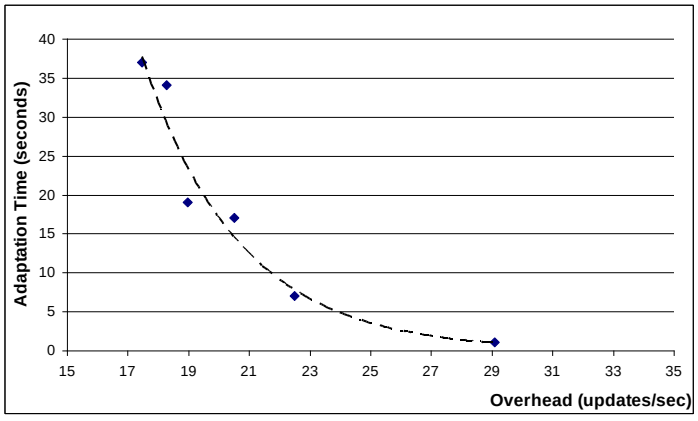


Figure 4-10: Adaptation time as a function of the incurred overhead for a network of 200 nodes

We have also analyzed the overhead in steady-state for different aggregation trees for networks of different sizes. Figure 4-11 shows the measurement results. The y axis represents the maximum number of processed updates across all nodes. The x axis represents the number of aggregating nodes in the tree, i.e., the degree of decentralization. Every point in the figure corresponds to one simulation run.

We observe that as we increase the degree of decentralization, the overhead decreases. For low decentralization degrees, the overhead decreases faster than for high decentralization degrees.

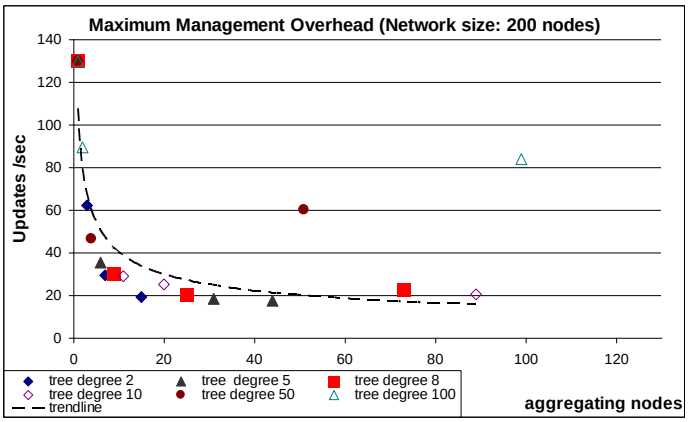


Figure 4-11: Management overhead incurred by A-GAP as a function of the number of aggregating nodes (degree of decentralization) for different aggregation tree degrees

From these results, we conclude that the overhead can be significantly reduced adopting configurations with a low degree of decentralization. For instance, for a 200-node network, using just 7 aggregating nodes (less than 5% of the nodes in the network) the overhead can be reduced by more than 75% compared to using one aggregating node (i.e., using a centralized approach).

An interesting observation is that these results suggest that the overhead is almost independent from the tree degree for balanced trees of constant degree. This holds for all our experiments except those where the tree degree is comparable to the network size (for tree degrees of 50 and 100 in the case of a 200-node network. This shows in the figure, where two points, corresponding to tree degrees 50 and 100, do not fit into the negative exponential trend). This suggests that (in terms of protocol overhead, and for balanced trees of constant degree) it is more important how much to decentralize, rather than how. In other words, the most important characteristic of the aggregation tree is the number of aggregating nodes in it.

In addition, our results show that the overall management overhead scales logarithmically with the number of aggregating nodes [70], which makes A-GAP scalable. Finally, our results show that the adaptation time primarily depends on the height of the aggregation tree [70]. The adaptation time remains short for low tree heights, and then it rises sharply with growing tree height. (In our experiments, this rise happens at the height of 3 or 4 levels.)

4.2 Real-time Monitoring for Routing

Real-time monitoring can play an important role in supporting the routing operations in clean slate networks. In fact, in many routing protocols and applications there are embedded operations related to the network state monitoring. Typically these monitoring functions are relatively simple and related to resource or topology discovery and resource consumption. Topology or dynamic path discovery is typical for reactive routing protocols (AODV [86], DSR [87]) while link quality evaluations are typical for so called link state routing protocols (CGSR [88], DSDV [89], OLSR [90], TBRPF [90] and OSPF [92]). Typical simple approach which is used by many protocols is based on periodic sending of HELLO messages. They are used for topology discovery, fault management and link state quality evaluation.

The novelty of the proposed approach in 4WARD is a separation of monitoring from routing. There are several advantages of the proposed approach:

- real-time monitoring functions are useful not only for routing but also for other purposes like anomaly detection, denial of service attacks, intrinsic network adaptation, etc. Due to the proposed approach the monitoring data are reusable,
- as opposed to relatively simple monitoring algorithms implemented in routing protocols in-network management monitoring operations are much more sophisticated and they

can provide network state information in more cost effective manner and with a greater accuracy,
- separating monitoring from routing make the routing protocols simpler and simplifies routing protocol operations, they may focus on core routing operations like maintenance of routing tables,
- the monitoring functions might be tailored to every networking technology i.e. they can deal with monitoring of physical and link layers in a cross-layer approach making core routing operations less dependent on the networking technology used.

Generally the INM monitoring operations can be split into two groups:

- Resource monitoring - typically encompasses: resource discovery, measurements of available resources quantity and quality, and failure detection.
- Traffic monitoring – operations which provide information about resource consumption and traffic properties.

While the traffic monitoring is widely used the measurement of resources is a new element. This is caused by fact that many latest networking technologies like xDSL, 802.16 and 802.11 in response to SNR will change modulation and channel coding schemes used, and in result link bitrate and delay may change. So, the classical assumption that link capacity is fixed is no longer true in these networks.

Traffic monitoring has at least two important motivations. The first one is to calculate the metric of links – this operation is of premium importance for proactive link-state protocols. The second one is to analyze the incoming traffic in order to identify flows or other specific traffic properties (for example applications which were responsible for traffic generation). The second approach is important in so called flow aware routing and application aware routing (classical approach of Deep Packet Inspection to routing).

From the management point of view, today's IP networks can be divided into two categories:

Well managed networks: This category includes all wired and wireless networks which can be seen as are deployed by network operators and deployment process is usually supported by careful planning procedures. During operation, such networks are managed and monitored by classical centralized OAM systems which are typically tied to the technical solutions used for every OSI layer (e.g. for SDH transmission a subsystem). Typically, there is no cross-system (cross-layer) information exchange between such systems which can lead to inefficiencies in overall system operation. The well known drawbacks of such a strategy can be observed in handling alarms in IP networks built on the top of ATM over SDH networks. It should be emphasized that in such 'classical networks' the topology is virtually fixed. Thus, after the deployment no network configuration is needed that poses sharp real-time constraints except the case of handling the failures (rerouting). The other exception can be linked to the increase of traffic which may lead to link overloading and call for link reconfiguration or traffic handling policy changes; this, however, can be pre-planned to a great extent in most practical cases. It is worth mentioning that in this class of networks the quality-of-service provided by layers below the network layer is generally very high, and so is the 'stability' of resources that have impact on routing. Thus, network resources which are available to the services are not changing in time. The latter property strongly impacts the design of routing protocols for this class of networks.

Ad-hoc networks (nomadic networks, etc.)**:** In this case, typically there is no infrastructure nodes nor links, the topology of the network can change very fast, and the quality of links varies in time (in terms of available bit rate, packet error rate, link availability, etc.). These features make classical routing protocols insufficient to guarantee satisfactory operation of the network. In response to this problem a plethora of routing protocols has been developed by IETF MANET group (for example AODV, OLSR, DSR). Many of them have incorporated some kind of real-time management and monitoring used e.g. for discovery of network topology, link quality estimation in case of proactive routing, and building the forwarding tables. Monitoring is also used for efficient mobility handling. It has to be emphasized that monitoring operations in

all MANET protocols are not separated as a set of specific functions orthogonal in a sense to the routing and data forwarding functions.

Both classes of routing protocols as above are typically not tied together and they are still designed and analyzed separately despite an enormous growth of networking devices that begin to be used in mixed wireline/wireless heterogeneous (computers, multimedia terminals, multifunction mobile phones, wireless access points, small IP switches, home networks) environments". This growth makes centralized, end-to-end management difficult, and sometimes impractical.

Additionally, there is a significant growth in deployment of networking solutions which have intrinsic limitations in providing reliable and stable transport service. Notably WiFi devices which work in unlicensed radio bands (ISM) may experience unexpected link quality deterioration as a result of radio interferences from other devices. The built-in auto fall-back mechanism, which adapts the maximum data transmission rate to the radio channel quality (SNIR), impacts available link capacity in unpredictable manner. The mixed wireline/wireless network can be treated as new kind of access network with a high potential especially for nomadic users. It is characterized by three important properties:

- The management operations are limited due to intrinsic technological limitations and the scalability issues.
- The available resources of nodes and links can slowly change in time - they are not fixed anymore.
- The network topology of this type of networks is not fixed.

Accounting for these properties calls for a new approach to the routing problem. The basic requirement here is that the routing protocol has to take into account the extremely unpredictable and uncontrollable (in a sense) behaviour of network devices that typically leads to a limited reliability of the network. So, the role of routing protocol is to create reliable routes in a dynamic manner taking into account the factors as above. Multipath routing techniques seem to be a useful tool which can help to achieve this goal.

Real-time monitoring plays a crucial role for this type of routing – it should measure the resources available as well as their usage. Separating RTM operations from routing provides a new approach to protocol engineering – 'core' routing operations are independent from RTM operations.

In the network with limited management (as described above) real-time monitoring (RTM) can play multiple roles:

- It can be used for network topology discovery and network topology changes.
- It should monitor the quality of the offered resources using a set of appropriate metrics.
- It should monitor the usage of the network resources.
- Based on anomaly detection (at the traffic level) it can discover link or node failures.

In order to perform its operations not only RTM function has to be implemented in all network nodes but also it should allow for distribution of monitoring data among the nodes. In order to guarantee robustness of the network (no single point of failure) the RTM subsystem should be distributed, nodes should have self-management properties and the exchange of information between nodes should be minimized in order to provide low control data overhead.

The proposed use of RTM for routing leads to novel decomposition of routing architecture. The decomposed architecture of routing is presented in Figure 4-12. It consists of the following modules:

- Real-time monitoring subsystem (RTM) – a component of INM.
- Routing Table Building Module (RTBM).
- Data Forwarding Module (DFM).

This decomposition enables different approach to routing compared to that typically employed – the routing protocols don't have to measure the quality and the quantity of offered resources

nor their usage. This is the role of RTM. The other modules have only to analyze the information obtained from RTM via API. This information is used for forwarding table creation and data forwarding. Using this information various algorithms can be deployed, including adaptive routing algorithms sensitive to network state, and multipath routing algorithms, the latter being emphasized as important element of 4WARD (developed in WP5). The proposed decomposition can facilitate a new approach to routing where every functional block (subsystem) can be optimized or modified independently of the others.

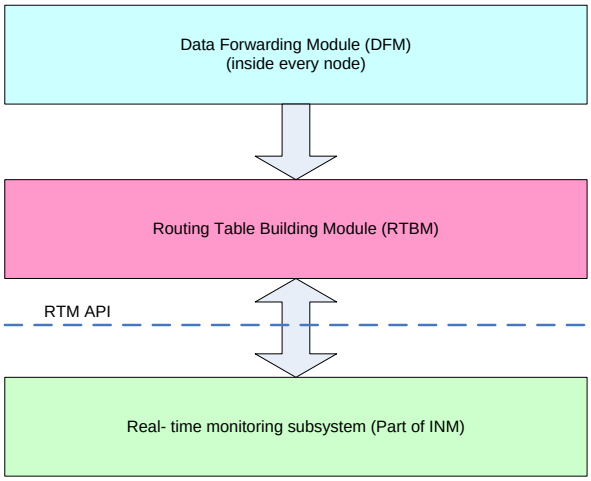


Figure 4-12 Generic decomposed architecture of routing subsystem in 4WARD networks

Every module of the architecture has implemented a set of specific mechanisms or functions. RTM of course serves for multiple purposes but it offers mechanisms which support routing operation. The following functions are performed by RTM, RTBM and DFM:

RTM Subsystem

The RTM subsystem is responsible for collecting and disseminating information about available and consumed network resources. This information is distributed among the nodes. This dissemination can be coupled with some processing (aggregation) in order to limit control data overhead and improve scalability of the routing system. Information about the network state can be obtained *via* active probing and passive measurements (traffic observations). Additionally, the network conditions in a near future can be predicted via analysis of the measurement history. The routing procedures may have impact on the frequency of issuing active probes, sampling the traffic or calculating traffic statistics. In order to limit the overhead traffic induced by RTM, the measurement reports consist of averaged data and during the dissemination they are subject to further reduction in intermediate nodes. In several routing approaches there is a possibility of reducing the accuracy of the information which describes the network state or aggregation of information about resources which are mutually distant one from another. This approach is especially suitable in case of the so-called adaptive loose source routing approach (no strict paths between sources and destinations). Active probing of resources enables network topology discovery and learning topology changes, and it also can be used for early failure detection. It is worth mentioning that intrinsic RTM mechanisms enable the discovery of local topology of the network, and even of local connectivity structures (in terms of n-hops); clearly, RTM is not responsible for creating end-to-end paths. RTM resource measurement reports consist of information about the network, node or link state according to a predefined metrics. This metrics typically include information about [60]:

- links/paths delay,
- links/paths delay jitter
- links/paths offered capacity,

- links/paths usage level,
- links/path reliability/ reputation,
- links/paths BER/PER,
- nodes power reserve (important in case of battery powered nodes).

The parameters listed above are typically technology-independent, however, wireless and wired networks require different approach to resource consumption estimation (even given the same metric). Additionally, in wireless networks the parameters of different OSI layers will often be used for metric composition (cross-layer approach).

In 4WARD it is expected that RTM will provide basic metrics parameters which will be subsequently processed by the routing procedures.

RTM control data is exchanged with RTBM via API. It is envisioned that there can be a mutual data exchange because some operations related to the end-to-end path monitoring (or monitoring of generic paths) can be incorporated into RTBM. The mutual exchange of the measurements between RTBM and RTM will be investigated later on.

Routing Table Building Module (RTBM)

This module takes the information about all available/discovered resources and local (n-hops) connectivity table via API from RTM subsystem in order to create end-to-end routing tables. In case of multipath routing several paths between a source-destination pair are created. This operation can be proactive (i.e. paths are established before the traffic is send) or reactive (path setup on demand). The existence of RTM local and overlapped topologically connectivity tables helps in fast setup of end–to-end paths. The detailed definition of the operations of this module will be due to WP5. It is expected that a plethora of a variety of routing protocols can be implemented as RTBM, dependent on the network type which is used and the type of the service offered.

Data Forwarding Module (DFM)

This module is responsible for forwarding the data using the information about paths from routing tables which are created by RTBM. The forwarding schemes may take into account QoS requirements by splitting a data flow among different paths in order to increase overall data throughput and/or resilience. Again, the detailed definition of operations of this block will be due to WP5.

The Generic Path (GP) constitutes an important 4WARD paradigm. WP5 is still working on the definition of generic paths [5]. In the context of the proposed architecture for routing GP can be treated as a service which is using physical resources monitored by RTM subsystem. It is expected that every GP will have built-in monitoring (as a part of service quality monitoring) which may cooperate with RTM in order to increase the resource monitoring accuracy and reducing control data overhead. Figure 4-13 shows the envisaged relation between RTM and other modules from Figure 4-12 and the generic path.

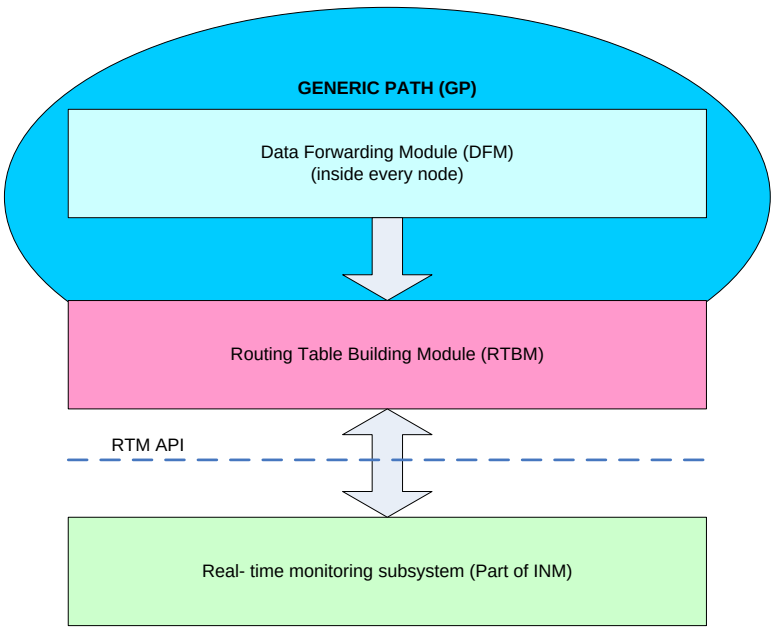


Figure 4-13: Generic Path and architecture of routing subsystem of 4WARD networks

In the proposed architecture every GP uses an instance of a specific DFM and uses a subset of forwarding table created by RTBM (in a VPN style). This draft proposal requires more detailed analysis but is in line with current WP5 vision.

4.3 Use of Real-time Monitoring to Create Situation Awareness

INM will introduce new qualities into the network. The goal is the automation of management tasks by enabling self-management and learning capabilities in the network. Decision-making processes are moved into the network and executed in the network nodes. In order to achieve such level of intelligent behaviour it is necessary to enable cognitive features in the nodes. Instead of merely reacting in pre-conceived ways, the system is capable to respond even to previously unencountered situations and becomes open to evolution.

Situation awareness provides here the appropriate conceptual framework under which advanced self-management capabilities can be designed. Situation awareness addresses the observation and collection of network state data, the interpretation of its meaning and understanding of the implications of the network situation. The objective is pre-emptive strategic behaviour. The network is not just observing what has already happened, but aims to derive what is expected to happen in the immediate future. By detecting critical developments at an early stage the system will initiate countermeasures that prevent undesirable situations to actually occur.

A definition of the conceptual framework for situation awareness and its integration with the 4WARD self-management architecture are presented. Benefits of the situation aware management will be demonstrated in application of the concepts to various management problems. Focus is on anomaly detection in network traffic, support for dynamic routing adaptation and congestion control in the network.

4.3.1 Definition of Situation Awareness

Situation Awareness is a conceptual framework originating from research in human factors. Situation awareness is explained as “being aware of what is happening around, understanding how information, events, and own actions impact goals and objectives now and in the near future” [104]. Situation awareness (SAW) is a capability required by an actor operating in a challenging environment. Changes in the environment demand constant observation of events, comprehension of their meaning, projection on probable near future outcomes and decision making for own actions in order to optimally adapt and react to the current situation.

Wrong decisions may lead to drastic consequences possibly resulting in loss and severe damages, and inadequate reaction to the situation may ultimately even lead to disaster.

Typical application fields include e.g. aviation and air traffic control, control of critical infrastructures like nuclear power plant management, emergency situations, military C3I (Command Computers Control and Information) etc. But also operator networks represent critical infrastructures, and network management tasks are characterized by the same constraints and challenges. Each of the classical FCAPS management categories implies in itself decision making processes that depend on situational understanding.

By executing management functions on the network node, management tasks can be automated and performed more efficiently. In order to realize such an approach it is necessary that the network nodes themselves execute monitoring control loops and perform management decisions on how to best adapt to the dynamic environment. This requires situation awareness: the node must observe what is happening, must understand the relevance of perceived events and must select actions that are in line with the given management policies and are expected to best achieve the set management policies.

4.3.2 Situation Awareness Model for INM

The conceptual model for situation awareness places itself within the self-management control loops. It identifies factors that influence the creation of situation awareness and its relationship to the decision making process.

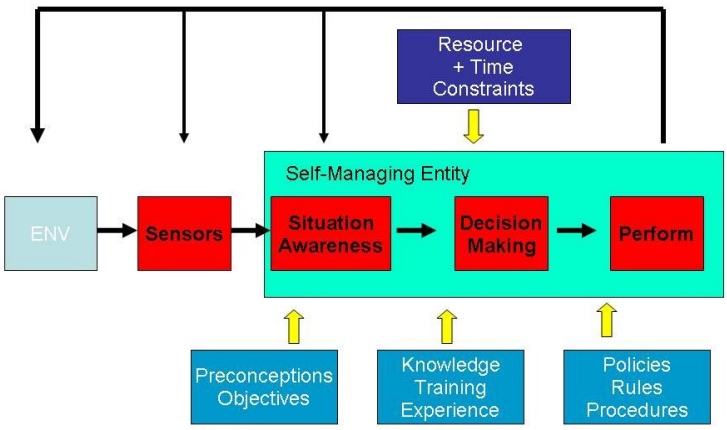


Figure 4-14: Situation awareness in the self-management control loop

Figure 4-14 depicts the self-management control loop. The self-managing entity continuously monitors its environment and reacts to the perceived observations: sensors and monitoring provide the observational raw data. Situation awareness is then the first step in the decision making process. Out of the raw data the current situation is extracted and evaluated. This information is input into the decision module that has to select the best response under the current situation. The execution module performs those actions that adapt the entity to the environment and attempts to influence the environment by their feedback in a favourable way. By self-learning, however, also the control process itself is adapted. The situation may influence and modify the data collection and evaluation process of the data.

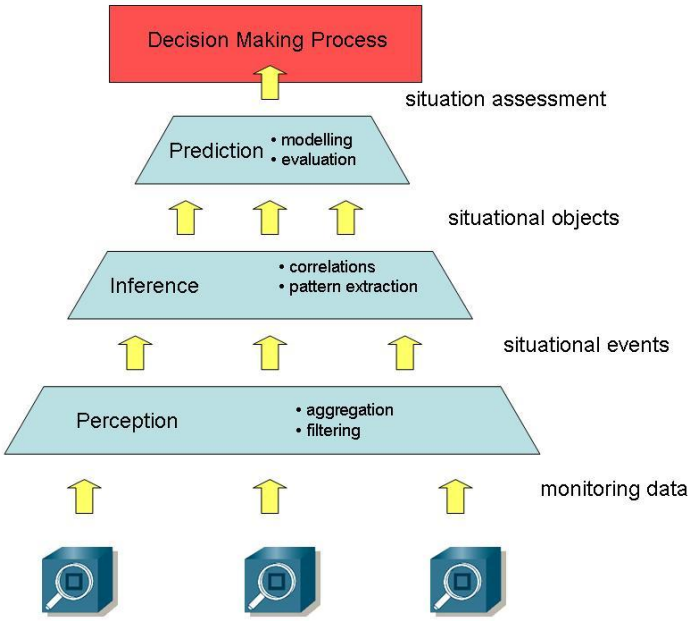


Figure 4-15: Levels of situation awareness

The problem of extracting the current situation out of the observations can be broken down into the three stages of perception, inference and prediction, see Figure 4-15. At the first step the raw observational data has to be consolidated. Filtering and aggregation allows to reduce the volume and to prevent information overload by improving the noise to information ratio. The process of perception may include mapping between formats and tagging with meta-information like time-stamps, observation point identifier, precision etc. The result of the basic perception process is consolidated situational events that are further analyzed in the next step of inference. Those events are correlated and lead to an understanding of a given situation. The situation is then assessed in order to derive a prognosis on what is going to happen next. This information is input into the decision making process in order to take actions.

The goal is to determine how the situation evolves and detect critical developments at early stages. Furthermore, prediction techniques help to assess how different alternative decisions would influence the situation.

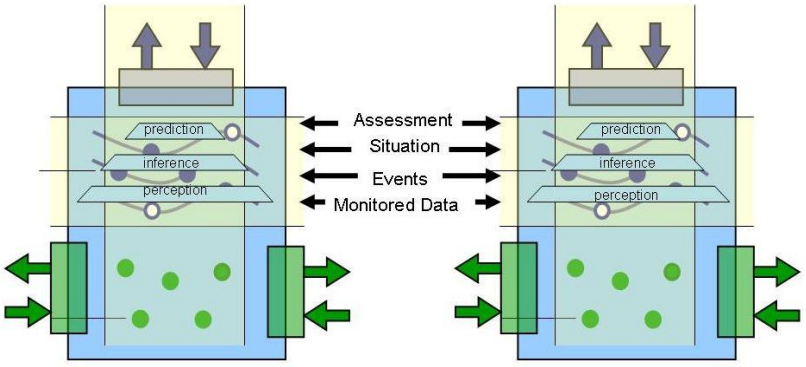


Figure 4-16: Sharing of situational information

Nodes self-monitor in order to collect information as a basis for local decisions. The process of decision making can be enhanced via communication and information sharing (Figure 4-16). Sharing of information can refer to data from all stages: the raw monitoring data, consolidated situational events, situations objects, but also assessment of the situation and results of the local decision process for cooperative decision making.

Furthermore, the observation of the behaviour of neighbour nodes assists in assessing their integrity, e.g. whether they behave in accordance to their assigned roles, whether they origin

or transfer malicious traffic, whether they are overloaded, etc. This information helps to weight information from neighbours according to trustworthiness, reliability and relevance.

Entities and messages that are involved in the process in establishing situation awareness are depicted in Figure 4-17. Data Providers and Data Sources register themselves via an interface called Data Provider to the Directory, which is the key instance of the whole process as it is a mediator to the offered location information. Data Aggregators that are interested in situation information send a resolve to the Directory requesting the location of this situation information. Typical Data Aggregators that should be mentioned here are anomaly detection, route optimization / adaptive routing, congestion control and distributed validation. This way, from the INM architectural point of view a Data Aggregator could be interpreted as an INM application. Upon receiving the location information (which is similar to an URI) the Data Aggregator requests the situation information via its interface Data Consumer either by sending a “get” or “subscribe”. In the former case the corresponding Data Provider will just reply once with the requested information, in the latter it will send this information continuously to the requesting Data Consumer every time new information are available.

To get a better understanding of this process each entity and message is described below Figure 4-17 in more detail.

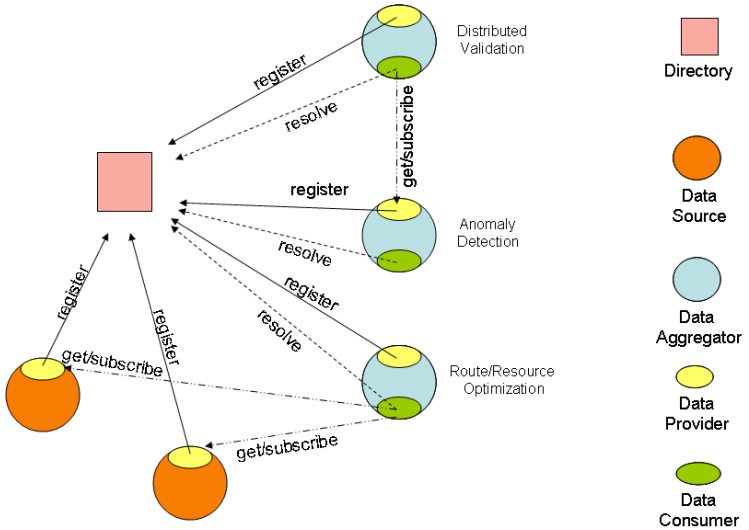


Figure 4-17: Situation awareness framework

Directory: The Directory is an entity that has knowledge about the location of offered information. It will reply to resolve requests for such information (which are known and registered at the directory) with an URI equivalent indicating the location of that information

Data Source: A Data Source is an entity (e.g. sensor) that collects low level information such as traffic load and registers itself at the directory to offer those information to other entities (could be also seen as situation raw data)

Data Aggregator: A Data Aggregator is an entity that uses information such as load, BER, etc., aggregates them (e.g. by applying certain algorithms), acts accordingly and offers those information to other entities via the directory. Example for INM applications are anomaly detecting, resource optimizations, distributed validation, etc.

Data Provider: A Data Provider is an interface at Data Aggregator/Source which offers collected information to requesting entities (e.g. via IPFIX or XML)

Data Consumer: A Data Consumer is an interface at a Data Aggregator/Source which requests the location of information at the directory and downloads those information from the indicated location afterwards (e.g. via IPFIX or XML)

Register: A Data Provider registers itself at the Directory with an URI equivalent indicating the location of its offered information to other entities or components

Resolve A Data Consumer that wants to get certain situation data sends a resolve to the directory requesting the location information of this data. The reply sent by the Directory to this resolve request will contain an URI equivalent with the requested location information.

Get/Subscribe A Data Consumer that is aware about the location of the needed information (a resolve request for the UCI has been sent previously) either sends a get or subscribe to the corresponding Data Provider. In case a "get" is sent to the Data Provider it will just respond once with the requested information. Otherwise if a "subscribe" is sent to the Data Provider it will transmit the requested situation data continuously to the Data Consumer if new information are available. It's worth to note that from the Data Consumer's point of view it does not matter whether the Data Provider is located at a Data Provider or Data Aggregator.

4.4 Traffic Matrix Estimation

Internet service providers need to know end-to-end traffic demands to analyse the load on their platform and to optimize the network. The traffic matrix is composed of the traffic demands (e.g. in Mbit/s) between all pairs of edge nodes (routers) within a considered topology. We have designed a method to estimate traffic matrices. Origin-to-destination traffic flows are calculated from statistical data measured locally by the routers.

Routing of data within a domain under unique administration is usually based on interior gateway protocols following the shortest path first (SPF) principle. For a long time, link load statistics were the only standard source of measurement in networks to observe the traffic development, in order to upgrade links when the load approaches a threshold and to estimate the flow demands. Today, technologies support traffic engineering and network planning, allowing setting up predefined routing paths in normal operation as well as backup paths for eventual failure situations. In addition, the traffic volume on each path can be obtained from standard MIB values provided by routers.

This provides the required flexibility to construct optimized path designs for traffic engineering purposes. Load balancing on links of the network can be achieved, which improves the overall quality of service (QoS) properties or can be used to increase the overall traffic load up to a level that still satisfies specified QoS requirements. If shortest routes are overloaded, some demands may be delivered on detours in order to balance the traffic flows to make full use of the available capacities.

For network planning, traffic matrices on a point-of-presence (PoP) basis are required. They enable the simulation of various extension scenarios and the decision on topology changes. When traffic between two nodes exceeds a certain level, a direct link between those nodes might be more efficient than the extension of existing links, which would imply a higher percentage of multi-hop traffic.

Our solution for the estimation of traffic matrices is presented in detail in Annex A.2.4. The presented method only requires as input metrics that can be computed locally by each router.

Currently, we can estimate the traffic matrix every 15 minutes. Our future work includes reducing this time interval. This would permit to achieve faster reaction times to changes in the networking conditions.

4.5 Distributed Cooperative Anomaly Detection

Within this project we are developing methods for anomaly detection and fault localization that can be used to both detect abnormal behaviour, and localize the source of the anomaly in the network. In future networks, distributed methods for autonomously detecting anomalies and network faults are essential to maintain critical functionality within the network.

Related work indicates that anomaly detection and fault localization are often based on network traffic analysis, such as traffic profiling, signature matching, signal analysis, statistical

analysis, etc (e.g. [61]-[66]). Apart from analyzing network traffic, another technique for anomaly detection and fault localization is *active probing*. For example, Rish et al propose a method that uses probe selection in order to detect and localize network faults [66].

Many of these methods are developed for centrally managed networks. However, in a highly dynamic environment, in which heterogeneous equipment and software is used and where the network topology is constantly varying with nodes being added, removed or replaced, distributed methods for INM are more convenient. The focus on autonomous INM means that a distributed, localized approach to fault detection and localization is preferable, and since we cannot rely on manual configuration of parameters, the algorithm needs to be able to autonomously adapt to network properties. Keeping these requirements in mind, we have developed a method for collaborative and distributive identification of network anomalies and faulty nodes and links, using simple end-to-end test transactions or *probes*.

The approach to local link monitoring used here is somewhat similar to that used in some routing protocols such as BGP, but the purpose differs in that we want to resolve errors to link and node level and report these as efficiently as possible for all entities in the network, while autonomously adapting to a wide variety of link qualities, including e.g. very lossy wireless channels. Our approach is more similar to heartbeat monitoring for fault localization in distributed systems (e.g. [67]) or local testing for fault detection (e.g. [68]), but those methods do not resolve link or node failures or adapt fully to local conditions. In the approach that we propose, statistics is applied in order to detect and reduce the uncertainty of a potential fault. Further, the algorithm adapts to varying network conditions which increases the robustness to e.g. false alarms.

Our approach to distributed fault-management is to perform necessary operations in two steps. In the first step, abnormal network behaviour is detected using local measurements at each node, aggregate measures from subnets, and probes, in order to test the accessibility on network level of links and nodes, or the availability of services on the application level. The second step is initialized whenever a node has detected a possibly anomalous node in the first phase, in order to localize the fault to a certain node, process or link.

For the developed algorithm, we assume that each node can perform a simple end-to-end test transaction, such as an ICMP request, on its neighbours, and measure the latency of this request. Further, each node needs to know the local topology, which in this case means awareness of all other nodes within two hops. Whenever a new node connects to existing nodes, it announces its presence and receives information about all nodes within two hops. The existing nodes that the new node is connected to notify all nodes about the added node, such that all neighbour lists are kept consistent. How this can be done in terms of the framework architecture is described in Section 3.7.1.

If the local topology is known, simple test transactions are done to perform active probing for detecting faulty components and links. Each node probes its neighbours at time intervals that are adaptively determined by probe reply delays measured on the link to which the neighbour is connected. This way the link load with respect to probe traffic is reduced, compared to other methods with fixed probing intervals. Using the simplifying assumption that dropping a request is independent of the time to get a response Δt, the probability of getting a response to the probe within Δt is

$$P(R_{\Delta t}) = (1 - P(D)) \int_0^{\Delta t} P(t)dt \quad (1)$$

where $P(t)$ is the probability density of probe responses at t, and $P(D)$ is the probability of a dropped request. Further, we assume that all probes are statistically independent, which allows us to write the total probability of lacking a response given a set of probes $\{R_{\Delta t}^{(1)}, R_{\Delta t}^{(2)}, \ldots, R_{\Delta t}^{(n)}\}$ as

$$P(\neg R \mid \Delta t^{(1)}, \Delta t^{(2)}, \ldots, \Delta t^{(n)}) = \prod_{i=0}^{n} (1 - (P(R_{\Delta t}^{(i)})) \quad (2)$$

such that with each failed probe the probability of a response decreases. When the probability of not receiving a response given previous probes has reached below a predefined threshold, a fault has been detected, and the process of fault localization is triggered.

The aim of the fault localization process is to determine whether it is the probed node or the link that has failed. When a fault has been detected, the probing node will send a request for assistance to the neighbours of the possibly anomalous node, such that it can be determined whether the fault is in the possibly anomalous node or on the link. Upon request, each one of the assisting nodes sends probes to test the possibly anomalous node, reporting the result back to the requesting node. If any assisting node reports success, we know that we have a link fault. If no assisting node reports success, we can conclude that the node has failed. If only a subset of the nodes return success, the result is inconclusive. The procedure is illustrated in Figure 4-18: and Figure 4-19:

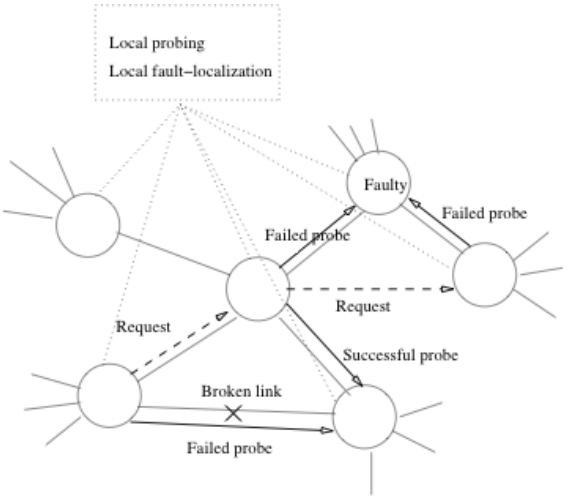


Figure 4-18: When a probe fails, the probing node asks neighbouring nodes for assistance in order to identify whether it is the probed node or the link that causes the anomaly.

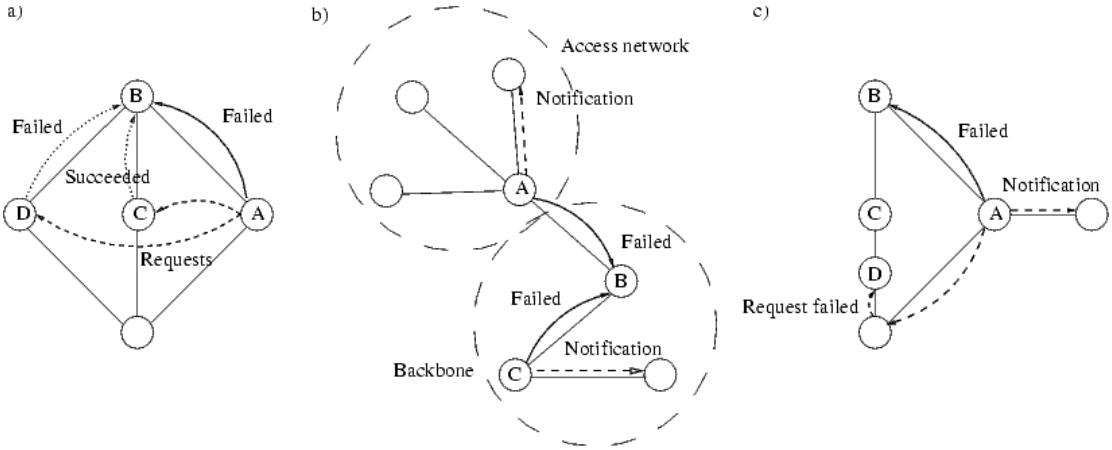


Figure 4-19: Figure a) shows the normal case. Node A cannot reach B and sends requests to C and D to test the accessibility of B. Since the probe between C and B is successful, the link between A and B is considered broken. In figure b), A cannot access B or its neighbour. In that case A directly notifies relevant recipients. In figure c) The neighbouring node C is inaccessible due to a faulty intermediate node on the path. The probing node A ignores this and notifies relevant recipients about the probed node, possibly including the closest node on the path to C, in order to reduce detection time.

More formally, let *n* be a node in the network. Each node needs to keep track of the set of neighbouring nodes, N_n, as well as the sets of neighbours to each of these neighbours *i*, N_n^i. The procedure taken when connecting or disconnecting a node to the network can then be described as in Figure 4-20 performed for both connecting nodes.

Algorithm 1 Connect node $\hat{n}$ to node n

Require: Valid nodes n and $\hat{n}$
  $N_n \leftarrow N_n \cup \{\hat{n}\}$
  $N_n^{\hat{n}} \leftarrow N_{\hat{n}} \setminus n$
  Monitor node $\hat{n}$ from n

Algorithm 2 Disconnect node n

Require: Valid node n
  for all $\hat{n} \in N_n$ **do**
    $N_{\hat{n}} \leftarrow N_{\hat{n}} \setminus \{n\}$
    for all $\tilde{n} \in N_n^{\hat{n}}$ **do**
      $N_{\tilde{n}}^{\hat{n}} \leftarrow N_{\tilde{n}}^{\hat{n}} \setminus \{n\}$
    end for
  end for

Figure 4-20: Algorithms for connecting and disconnecting nodes

Further, let each node *n* store an *error state* S_n^i for each neighbour *i*. Each S_n^i represents the current state of n_i as viewed from *n*, and can be assigned one of the following values:

- *No fault*. No fault has been detected in the neighbour.
- *Link or node failure*. A fault has been detected, but it has not been or it is not possible to determine if it is a link or node failure.
- *Link failure*. A fault has been detected and diagnosed as a link failure.
- *Node failure*. A fault has been detected and diagnosed as a node failure.

The fault detection and localization procedures can then be described as in Figure 4-21. Note that storing and making use of the current state of neighbours is not strictly necessary for correct operation of the basic algorithm. However, it does significantly reduce the number of

Algorithm 3 Monitor node $\hat{n}$ from node n

Require: $\hat{n} \in N_n$
  repeat
    if Test node $\hat{n}$ from n fails **then**
      if $S_n^{\hat{n}} = \textit{No fault}$ **then**
        for all $\tilde{n} \in N_n^{\hat{n}}$ **do**
          Confirm failure of $\hat{n}$ for n in $\tilde{n}$
        end for
        if Any $\tilde{n} \in N_n^{\hat{n}}$ report success **then**
          $S_n^{\hat{n}} \leftarrow \textit{Link failure}$
          Report failed link from n to $\hat{n}$
        else if All $\tilde{n} \in N_n^{\hat{n}}$ report failure **then**
          $S_n^{\hat{n}} \leftarrow \textit{Node failure}$
          for all $\tilde{n} \in N_n$ **do**
            $S_{\tilde{n}}^{\hat{n}} \leftarrow \textit{Node failure}$
          end for
          Report failed node $\hat{n}$
        else
          $S_{\tilde{n}}^{\hat{n}} \leftarrow \textit{Link or node failure}$
          Report link or node failure
        end if
      end if
    else
      if $S_n^{\hat{n}} \neq \textit{No fault}$ **then**
        $S_n^{\hat{n}} \leftarrow \textit{No fault}$
        if $S_n^{\hat{n}} = \textit{Node failure}$ **then**
          for all $\tilde{n} \in N_n$ **do**
            $S_{\tilde{n}}^{\hat{n}} \leftarrow \textit{No fault}$
          end for
        end if
        Report working link from n to $\hat{n}$ and node $\hat{n}$
      end if
    end if
    Wait τ s
  until $\hat{n}$ disconnects

Algorithm 4 Test node $\hat{n}$ from n

Require: $\hat{n} \in N_n$
  repeat
    Send test transaction to $\hat{n}$
    Wait θ s
  until Any response or $\prod_i (1 - P(R_{\delta t}^{(i)})) < \psi$
  if Any response **then**
    return Success
  else
    return Failure
  end if

Algorithm 5 Confirm failure of $\hat{n}$ for n in $\tilde{n}$

Require: $\hat{n} \in N_{\tilde{n}}, n \in N_{\tilde{n}}^{\hat{n}}$
  $t \leftarrow$ Test node $\hat{n}$ from $\tilde{n}$
  Report t to n

Figure 4-21: Fault detection and monitoring algorithms.

requests for assistance and thus messages in the network. Also note that monitoring of neighbours does not stop when a node or fault is detected as faulty, as we want to be able to identify if the problem goes away.

In the algorithm, we need estimates of the parameters for the distributions *P(D)* and $P(\Delta t)$. For the first one, we can easily estimate the parameters based on the recent fraction of dropped requests for the link. In the case of $P(\Delta t)$, we also need to select a suitable parameterization for the density function. Thus, we performed a number of studies of link latencies in both wired and wireless networks, resulting in the choice of a gamma distribution. Figure 4-22 and Figure 4-23 show examples of latency distributions along with the estimated gamma distribution for a number of wired and wireless links. The parameters of the gamma distribution are easily estimated from only a few counters for each link using a method of moments approach.

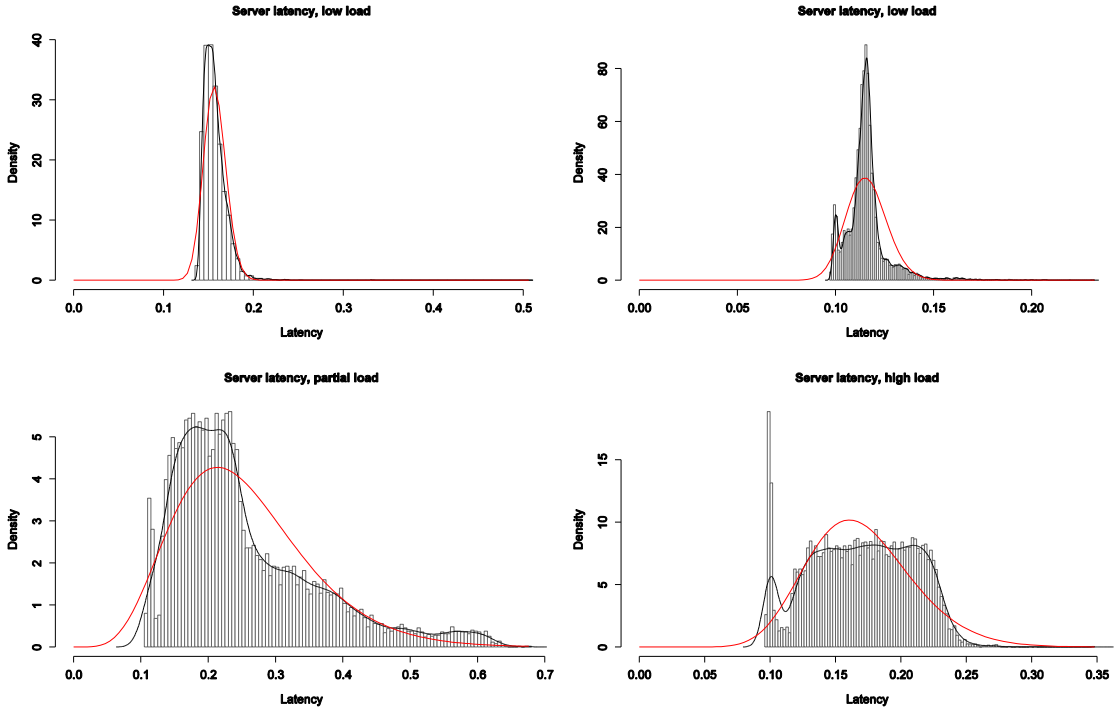


Figure 4-22: Examples of latency distributions on a wired network. Starting at the upper left, the graphs describe the latency distribution on a local Ethernet link on servers with no, low, medium, and high load respectively.

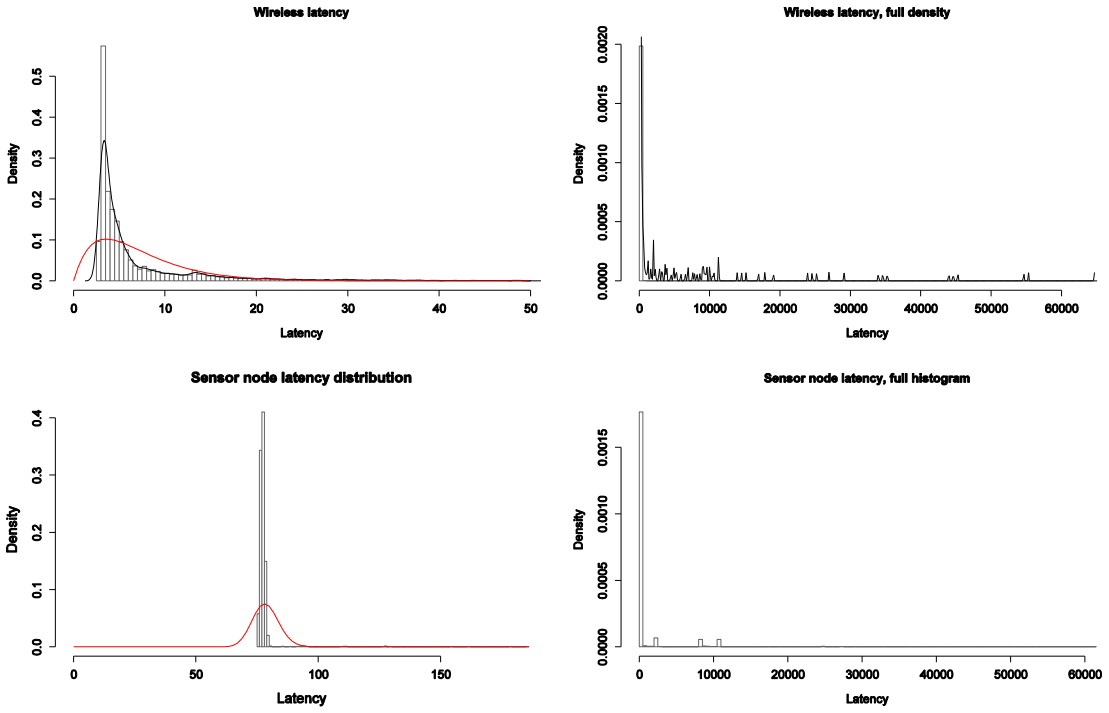


Figure 4-23: Examples of latency distributions on wireless networks. The upper two graphs represent latencies in a 802.11b network with very low signal strength, and the lower two the latencies in a very simple sensor node. The graphs to the left show estimated gamma distributions on truncated data, while the graphs to the right show (exaggerated) versions of the actual histograms, indicating that in both applications there are connection problems that will be reported as intermittent error in our algorithm.

For a completely adaptive, zero-configuration approach, we also need to set the timing parameter between probes τ and the fault-detection limit ψ autonomously. Although they ultimately depend on the actual cost of probing, we believe that we can set them to fixed values, using a multiple of the expected link latency for τ, based on experience from large-scale simulations on realistic networks that we are currently performing. Here we will assume that the cost of probing is proportional to the rate of expected link latency. In general, the cost is a trade-off between performance (in terms of detected anomalies) and link load.

The main benefit of our statistical approach is that the time interval with which a probe is sent and also the number of probes used to reduce the uncertainty of a potential fault are adapted to the measured link latency such that only a minimal set of probes are needed to detect an anomaly. This way, the traffic load on the link is minimally affected by probe traffic, compared to ordinary heartbeat monitoring in which probes are sent frequently at fixed time intervals. Since probe intervals are determined autonomously for each individual link, the need for manual configuration is significantly reduced while optimal monitoring performance is achieved. As both anomaly detection and fault localization are performed in a distributed manner our approach also scales well with the number of network components while adapting to local conditions. Our method is easily used in dynamic networks with varying topology caused by so called churning (i.e. addition, removal or replacement of nodes) - without manual configuration, nodes can automatically exchange information about themselves and their neighbours to other nodes without having to know anything about the network topology. Finally, network equipment of today already fulfils most of the requirements needed to carry out described operations - the implementation of the protocols above should therefore be relatively easy.

The algorithm has been implemented in the OMNET++, and we are currently performing tests on large-scale realistic topologies to test the performance of the algorithm. Initial results show

very fast response time to induced errors on both links and nodes, while the induced traffic is relatively low.

4.6 Lessons from Developing the INM Paradigm

In this chapter we have presented our work to date on real-time monitoring, anomaly detection and situation awareness. Based on this work, we present some lessons we have learnt regarding the development of the INM paradigm.

Performance control. We have shown that the performance of a distributed monitoring protocol can be controlled along different performance dimensions, including protocol overhead, protocol accuracy, and adaptation time. Furthermore, we have shown that it is feasible to control the trade-offs among these performance metrics (Figure 4-9). Sample trade-offs are estimation accuracy vs. management overhead and adaptation time vs. management overhead. For these trade-offs we have learnt that allowing a modest degradation in one performance metric can lead to a significant improvement in another performance metric.

Scalability. Our simulation results indicate that key performance metrics of our monitoring protocols grow sublinearly with the system size. This applies to (1) the accuracy achieved by gossip-based monitoring for a given local overhead and (2) the overhead by our tree-based protocols for a given accuracy.

Robustness. We have also shown that it is feasible to design management systems that are robust to topology changes (e.g., node failures). We have shown that this is feasible for two families of monitoring protocols: gossip-based and tree-based. Moreover, we have shown that the adaptation time to node failures is very short, a fraction of a second.

Model-based management. We have shown the benefits of model-based management in the context of real-time monitoring and anomaly detection. We have learnt that the development of analytical models is key to design monitoring systems that are controllable and achieve performance objectives. In the context of anomaly detection, we have shown that such models permit providing probabilistic guarantees on the detection of anomalies.

Feasibility. The results that we have achieved to data strongly suggest that it will be feasible to realize the INM paradigm and obtain the expected benefits.

5 INM Self Adaptation

This chapter discusses network self-adaptation in the scope of INM. The network itself takes proactive or reactive network management actions for the purpose of recovering a fault, avoiding a predicted fault, or optimizing the network operation, by changing the network configuration, the network setup, or resource allocation. Chapter 4 discussed monitoring and situation awareness, which are the basis for learning the current state of the network and its operation. This chapter makes use of this knowledge in order to take corrective actions.

Self-adaptation operations may take the widest scope possible, over the entire network. While some entities might implement self-adaptation mechanisms, they do so in a local scope, within their responsibilities. WP4 assumes full responsibilities for network-wide self-adaptation operations. Examples are load balancing (congestion avoidance), resource allocation and/or optimization, or fault recovery.

The discussion about Self Adaptation is arranged as follows:

Section 5.1 studies distributed network management benchmarking, i.e., the extent at which distributed processing is beneficial. Examining a few case studies, the research looks for the optimal point of INM distributed processing, considering the network environment and performance tradeoffs of scalability, robustness, adaptivity and overhead.

Section 5.2 presents the core self-adaptation schemes, addressing resource reservation, resource optimization, self-adapting wireless multi-hop networks, configuration planning, event handler, probabilistic management paradigm, and congestion control.

Finally, Section 5.3 compiles conclusions and lessons for a clean slate approach.

5.1 Benchmarking of Distributed schemes

Deliverable D4.1 showed that legacy network management systems are primarily centralized, that is, executed and managed from a single entity; additionally, they are mainly controlled by humans. It is clear that this arrangement is not scalable, and opportunities for automation are limited, as the use cases of D4.1 reported.

INM relies on distributed functions to enforce reliable self-management features in the future Internet. In order to implement such behaviour, some sort of distributed processing is required. The research area of distributed network management benchmarking studies the extent at which distributed INM effort is beneficial. We believe that there is no general answer for all network scenarios. For each network management task, a different level of distributed effort is favoured. This concept is shown in Figure 5-1.

Taking a bottom-up approach, we are testing a few network management test cases, and try to locate the "sweet spot"; that is, the optimal point for the extent of distributed effort, considering cost-performance tradeoffs. Both qualitative and quantitative metrics are sought.

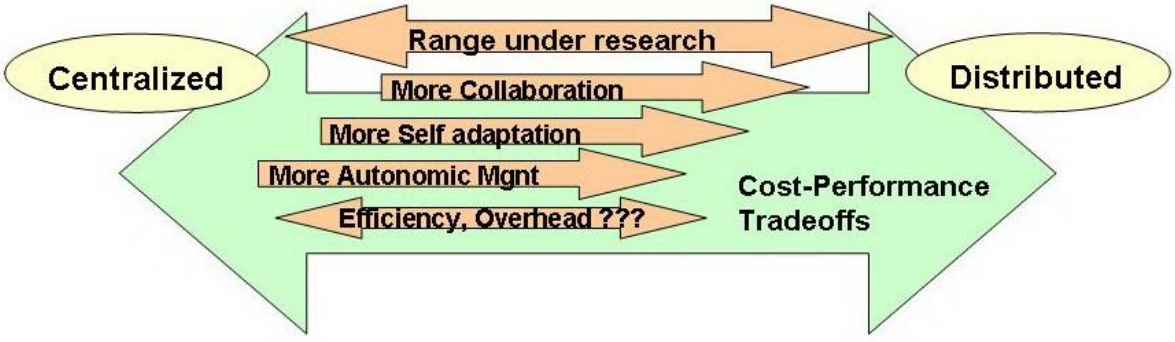


Figure 5-1 Distributed network management benchmarking

The 4WARD project requires us to innovate in two fronts:

1. Design a new network, the best we can come with, taking a clean slate approach;
2. Self-management, introducing a high level of automation in management operations

We believe that it is very difficult to successfully meet both objectives without any real-world experience in self-management. Consequently, the study follows a 2-stage approach:

1. Study of benefits and performances through the following steps:
   - Select a few test cases from existing networks, implementing centralized management.
   - Exploit distributed in-network management, compare the results with the legacy centralized method.
   - Establish benchmarks.
2. Design guidelines for a clean slate solution

Within 4WARD, we believe this is a preferred methodology to cope with the aforementioned complexity, which can significantly enhance the quality of our research.

We selected three network test cases to study

- Route protection for reliable networks (scenario 2 of deliverable D4.1).
- Adaptive data collection in sensor networks (scenario 1 of D4.1).
- Topology discovery (scenario 1 and 2 of D4.1).

The remainder of this section discusses each test case, and the conclusion reached.

5.1.1 Test Case 1: Route Protection

Route protection is a reliable mechanism, which reserves prearranged backup paths to accommodate fast link restoration with QoS. In the event of link failure, the backup link is immediately activated, with minimal service disruption. Such schemes are traditionally adopted in MPLS-based routing domains and are mainly used for multimedia applications, where IP-based routing recovery mechanisms are too slow to react. The protection mechanism is viewed as circuit-switched network emulation over packet-switched network and it can be applied to self-management capabilities of a generic path

Preliminary results [71] were presented at the IEEE infocom 2008 mini conference[1].

The Model

We studied 6 different protection schemes, ranging from fully centralized to fully distributed (Table 5-1):

- Global Recovery (GR): each primary path has one backup path; both paths share the end nodes only. Clearly, the setup and management of the backup link is centralized
- Unrestricted recovery (UR): each primary path may be protected by any number of backup paths. Moreover, each backup path may start and end at any point along the primary path, and may protect against failure of any number of elements. This scheme can be implemented in a partially-distributed manner
- Local Recovery (LR): a separate backup path is constructed to protect against a possible failure of each element along the primary path. Each backup path starts at the immediate upstream node of the protected element, and ends somewhere at the tail
- Restricted Local Recovery (RLR): as in LR scheme, the backup path starts at the immediate upstream node of the protected element, but ends at the immediate downstream node. This scheme is fully distributed
- Facility Local Recovery (FLR): backup paths are constructed as in the RLR scheme, however, a single backup path protects multiple primary paths
- Extended k-facility LR (KFLR): Same as FLR scheme, however, multiple (k) backup paths are arranged for protection of each element

[1] http://www.comsoc.org/confs/infocom/2008/tech_prog.html

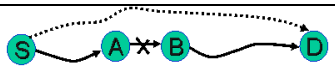
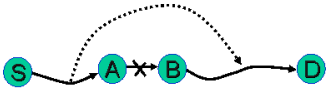
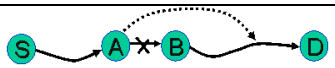
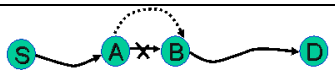
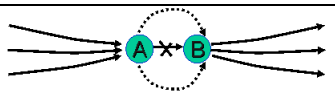
Recovery Scheme	
Global Recovery **GR** Centralized	
Unrestricted **UR** Partially Distributed	
Local Recovery **LR** Fully Distributed	
Restricted Local **RLR** Fully Distributed	
K-facility Local **KFLR** Fully Distributed	

Table 5-1: Protection schemes studied

We examined and compared the different protection schemes using the following criteria:

- Throughput maximization: a quantitative metrics, the primary criterion, used to signify the number of services that can be granted, given a fixed known network capacity.
- Time taken for path restoration: which protection scheme demonstrates the fastest path restoration, minimizing service down-time.
- Administrative overhead (a qualitative metric).
- Robustness: to what extent the protection scheme itself is prone to a single point of failure (a qualitative metric)

The study defined a formal model for the problem, modelling a capacitated directed network, comprising sets of source-destination pairs, with bandwidth demand and profit.

Results and Benchmarks

The hardness and approximation boundaries for all 4 variants of the problem were studied. We presented and implemented efficient heuristics for the various recovery schemes. We simulated the problem with sample data and analyzed the results, which are summarized in Table 5-2.

Recovery	GR	UR	LR	RLR	FLR	KFLR
Throughput	Better than LR Comparable w/ UR	Best	Better than RLR Comparable w/ UR	Worst		With K=2 equal to RLR
Time to restore	Slowest	Slow	Fastest	Fast	Fast	Fast
Overhead					Lower than RLR	Lower than RLR
Robustness	Single point of failure	Robust	Robust	Robust	Robust	Most Robust

Table 5-2: Results of route protection test case

- Throughput: expectedly, UR scheme shows the highest throughput, as it has no constrains on backup path formation. RLR scheme, being the most restrictive, demonstrates worst results. LR and GR schemes are comparable with UR
- Time to restore: LR is fastest, RLR, RLR, KFLR schemes are acceptable, UR is slow and GR is the slowest
- Administrative overhead: FLR and KFLR clearly demonstrate lower system-wide overhead, due to the fact that backup paths can protect multiple primary paths. The overhead of other schemes has to be further studied.
- Robustness: GR is prone to a single point of failure, where all other schemes are distributed in nature, and thus better in this respect. KFLR is naturally the most robust

Considering those benchmarks, LR scheme seems the preferred approach: it show throughput that is comparable with the best scheme, it is the fastest to restore, and is robust.

5.1.2 Test Case 2: Adaptive Data Collection in Sensor Networks

A sensor network has a limited amount of data collection resources. Depending on the specific situation which varies over time, there are several levels of interesting measurements (threats). It is desired that sensing resources are assigned dynamically in an optimal manner, in order to provide the best information for the specific situation.

Consider, for example, a monitoring surveillance system, with video cameras. Normally, the cameras will be arranged to cover the whole considered area. However, upon the detection of a threat in a specific area, close-by cameras should focus on the event, with more viewpoints and higher zoom, while still maintaining full coverage. Such change of focus requires dynamic allocation of cameras. Similar considerations are appropriate in other examples, such as network security systems, or sensor networks for monitoring environmental parameters.

Preliminary results [72] were published and presented at the IEEE ANM 2008 workshop[2]

The Model

We defined a formal model for the problem. The model includes a set of sensors with capacity definition (what can be measured), a set of targets with severity (what are we looking for), and a coverage matrix (which sensor can detect what area). We defined a profit function. Finally, we defined the profit maximization problem for the sensor-target assignment.

We examined and compared a centralized and a distributed approach, under pareto and uniform distribution of targets. We implemented a few centralized algorithms, and one distributed algorithm. For the distributed scheme, the network negotiates a local leader for each sensed target, following agent capacity negotiation with possible pre-emption, and concluding with the leaders' assignments of agents to targets. The algorithm is shown in Figure 5-2. ts_j denotes the time a target t_j is revealed for the first time. Leader message used to notify the agents about a target leader. Accept and preempt messages are used to respond with the availability of monitoring resources to cover a target. Monitor message is used for

The following DG scheme is run by each agent $s_i \in S, 1 \leq i \leq |S|$

1) When and if a new target t_j is found and $|Cover_i| < cap_i$
2) Start the guard timer
3) Multicast $leader(s_i, t_j, d_j, ts_i)$
4) Start waiting for $accept$ and $preempt$ messages with ts_i
5) If message $leader(s_k, t_j, d_j, ts_k)$ received
6) If ts_k is the earliest timestamp for t_j
7) Stop waiting for $accept$ and $preempt$ messages
8) If $gap_i > 0$
9) Send $accept(s_i, t_j, gap_i)$ to s_k
10) Else
11) Choose $t_m \in Cover_i$, such that $d_m = \min_d (Cover_i)$
12) If $d_j > d_m$ send $preempt(s_i, t_j, t_m, gap_i)$ to s_k
13) If $monitor(t_j)$ is received and there is a connection to t_j
14) Start tracking t_j
15) If any target t_m is preempted due to t_j
16) Forget timestamp for t_m and run DG for t_m
17) When the guard timer expires
18) If no earlier timestamp for t_j received
19) Stop waiting for $accept$ and $preempt$ messages
20) If $|A| \geq d_j$
21) Sort set A in the descending order of gap
22) Send $monitor(t_j)$ to first d_j agents from A
23) If $|A| < d_j$ and $|A| + |P| \geq d_j$
24) Send $monitor(t_j)$ to all agents in A
25) Sort set P in the descending order of gap
26) Send $monitor(t_j)$ to first $d_j - |A|$ agents from P
27) Forget all leader timestamps of two steps before current

Figure 5-2: Distributed algorithm for the adaptive data collection

[2] http://www.ee.kth.se/lcn/anm08/

final assignment of an agent to target coverage. For a specific agent, A and P denote the sets of received accept and preempt messages, respectively.

The research compared both approaches using the following criteria: coverage (what portion of targets were monitored), adaptivity, robustness, scalability, and administrative overhead. The coverage criterion was actually measured in the research, while the other criteria items are for consideration only.

Results and Benchmarks

We implemented efficient algorithms for both the centralized and the distributed control. Figure 5-3 shows the simulated results, comparing the coverage of both the distributed and the centralized algorithms. Table 5-3 summarizes the results.

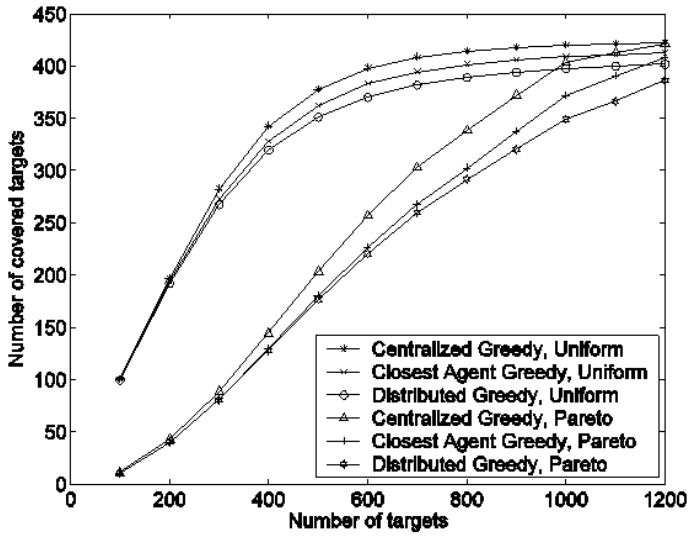


Figure 5-3: Number of covered targets, as a function of total number of targets

Method	Centralized	Distributed
Coverage	Higher	Lower (but comparable)
Adaptivity	Slow	Fast
Robustness	Low	High
Scalability	Low	High
Overhead	Low	High

Table 5-3: Results of adaptive data collection test case

Clearly, the distributed approach is highly adaptive, robust and scalable, important features of the desired solution. The lower coverage and the higher overhead are attributed to the rudimentary implementation: this is a first attempt for self-management, which requires further study and better implementation. We believe that an optimized implementation of the distributed approach can overcome those limitations, showing measurements comparable with the centralized approach, and thus, become the method of choice.

5.1.3 Test Case 3: Decentralized Topology Discovery

A key part of real-time monitoring is topology learning. Section 3.5.2 discusses the importance of topology discovery for INM. The process must be very efficient, in order to provide real-time topology information, with minimal/affordable overhead. Intuitively, some sort of distributed collaboration seems more effective, and we investigate this topic.

The Research

We examine the following current management examples, from which we extract guidelines for the clean slate approach, which will be applicable to the GP WP.

- A sensor network, with short-range wireless communication. Topology learning is required for access of any sensor to the gateway (root) or Mediation Points in general;
- A large switched Ethernet network, where the knowledge of local links is used to build a network-wide view.

As far as topology learning and distributed network management benchmarking, both examples are similar, and require the same solution. Switched Ethernet network is made of multiple inter-connected switches. Each switch maintains an Address Forwarding Table (AFT) for each port, which is continuously updated. AFTs provide information about addresses that are accessible from each port. We aim at learning the topology from AFTs, while minimizing the number of queries.

We employ 2 different algorithms:

- A centralized approach, in which every member of the Vertex Cover Group (VCG) is requested to send its AFTs to the management station, from which topology is learned. (VCG is a group of nodes from which all nodes and links are accessible)
- A distributed approach, where every member of the VCG resolves the topology around it. This approach can be taken incrementally; additional VCG members are involved only if the full topology is not yet learned.

Note that the second approach can possibly gather the full topology without traversing the whole VCG. VCG identification is an NP-Complete problem, and therefore, heuristics are used to identify a VCG that is not necessarily minimal. That means that the VCG has some redundancy, and the distributed process might be able to avoid it, thus learning the topology more effectively.

Results and Benchmarks

Preliminary results show that contrary to our expectations, the centralized approach is more efficient: it requires smaller number of queries for the topology learning process. As far as the amount of time for the complete learning, the distributed process is much faster, probably because portions of the full topology are discovered in parallel.

We can conclude from these preliminary results the following benchmarks:

- A tree-like structure lends itself better into a centralized approach; the complete information must reside at the root anyhow.
- Distributed effort exploits parallel processing, thus getting the results much faster; when speed of operation is the most important criterion, distributed topology learning is preferred.
- A centralized topology structure requires one or more "super peer" to manage the discovery process, and therefore is subject to a single point of failure. The decentralized approach is more immune, and is a better choice for emergency scenarios.

5.2 Self-Adaptation Schemes

The **I**n-**N**etwork **A**utonomic **M**anagement **E**nvironment, (I-NAME), is presented in Section 5.2.1, addressing resource management and resource reservation. Such environment enables the network entities to automatically detect the dynamically changing network's configuration, and react accordingly to the service's requests.

Section 5.2.2 researches collaborative protocols and algorithms for resource optimization in a dynamic network, primarily addressing network bootstrapping and network discovery.

Section 5.2.3 discusses a special network case, a situation-aware self-adapting wireless multi-hop network. Protocols, required functionality, parameters and metrics are addressed.

An innovative INM application is presented in Section 5.2.4, which pre-plan and verify configuration changes prior to implementing them. The application predicts the possible effects of such changes, thereby preventing undesired behaviour and potential failures.

Section 5.2.5 researches different paradigms for dissemination of information through events in a distributed environment, with no initial configuration. A generic model is proposed, supporting existing event correlation techniques, while addressing performance tradeoffs of overhead, timeliness, success rate. Design guidelines are derived for INM event distribution.

A probabilistic management paradigm for solving some major challenges of decentralized network management, is proposed in Section 5.2.6.

Pulse-coupled oscillators, traditionally used for time synchronization in ad-hoc networks, are innovatively applied to congestion control in packet networks, and presented in Section 5.2.7. Such congestion notification scheme is based on emergent behaviour, which can show faster response time in a scaleable manner, without additional configuration or management processes.

5.2.1 INM Environment for Resource Management and Resource Reservation

Delivering multimedia traffic with QoS guarantees over multi-domains is a major challenge nowadays. The domains can be composed by more than one access network, each with its capabilities, devices and users. In this context, we defined I-NAME as an environment performing in-network management and resource reservation tasks between network domains. Working with profiles (aggregated parameter sets), this is flexible to the network conditions and users requests. It was designed as an end-to-end solution for managing resources, overcoming and accommodating different types of users (in terms of network technologies – access and transport). Hence, it is not a signalling mechanism because it reacts to the environment context, assisting the functional components (such as ForMuX, Vnet) to offer alternatives for a given session. I-NAME is not a routing mechanism, but it assists the control plane in order to detect the best path to the destination. This is done based on the information objects collected and delivered to each strategic node through exchanged profiles. I-NAME was designed based on the clean slate node architecture, acting within INM Application level only, but relying on functional components like QoS and/or Routing Modules present in any node (at both INM kernel and INM Application levels).

I-NAME (In-Network Autonomic Management Environment)

The scope of I-NAME is to exchange dedicated management information between nodes through GPs specially established for inter-management. It will not actually reserve the resources, but will deliver/collect the information objects to/from any node. According to 3.7.2 it is supposed that mandatory QoS and routing modules are present in any node. However, only the strategic nodes (usually located in mediation points) will be able to perform more complex operation such as assisting ForMux to do operational resource reservation and management within GPs. I-NAME implementation requires a set of messages exchanged between entities placed in INM Application and INM Kernel space. These messages could be called **profiles** if they are applied in QoS, but as I-NAME is a general environment, its applications are not limited to the example given hereafter. It is up to network designers to involve it in negotiations, neighbour discovery, anomaly detection, event handling etc. by simply changing the profiles with proper messages to be exchanged through dedicated management GPs.

I-NAME Application Example: Resource Reservation and Resource Management

I-NAME defines profiles as aggregate QoS parameter sets (throughput, delay, jitter, packet loss, etc.) and gives personalized access (from the user point of view) and optimized services (from the network/ operator perspective). I-NAME defines profiles, as the QoS parameter sets requested, supported, negotiated, and adopted in the message flow between the INM FCs. Considering this, I-NAME defines four types of profiles needed to determine the path cost: (1) **Requested profile:** includes application request for resources, generated by the SN (Source Node). (2) **Accepted profile:** expresses the destination's availability for resources announced in the requested profile and it is generated by the DN (Destination Node). (3) **Negotiated profile:** expresses the destination's possibilities for resources announced in the requested

profile and it is generated by the DN. (4) **Adopted profile:** contains the source's availability for resources announced in the negotiated profile and it is generated by the SN. I-NAME message flow is based on the interaction and decision according to the information included in the profiles. The profiles should carry the network's capabilities from node to node in the path to the destination, but only the SN and DN could choose between different profiles in order to have a common view of the application requirements. To demonstrate the I-NAME concept we have used the QualNet Developer 4.0 Simulator. The implementation is based on two applications running simultaneously in each node: a client (I-NAME Client Application) and a server (I-NAME Server Application). On the GP from the source to the destination the messages are passing through strategic nodes which recognize and further process the I-NAME message flow.

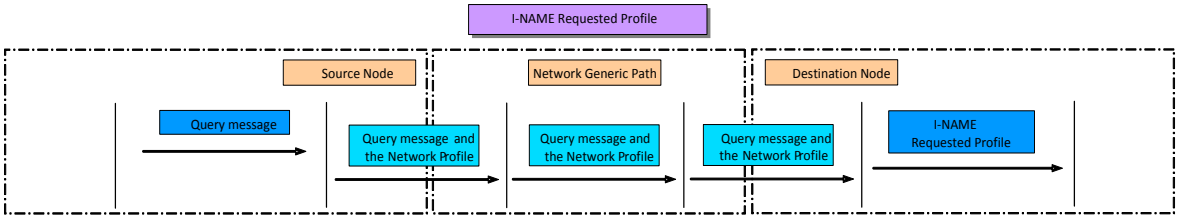


Figure 5-4: I-NAME Requested Profile message flow

To add a request, I-NAME Client Application sends a Query Message to the Server Application that includes the Requested Profile. The Server Application listens to a well known port and it stores a database including all the requests received. On the GP from the source to the destination, the network stamps in the Requested Profile its own capabilities. The message is define as Query message with network profile, which is the same as the initial Query message, plus the network's additional influence. This means additional delay and jitter by each network segment or bandwidth limitations introduced on that path. We call the path specific characteristics as the Path Profile. This could be modified by each network node, accordingly to the segment parameters passed by the Query Message through the path. Because of multiple paths, each time a new request reaches an intermediate node / strategic node, the Kernel forwards the best Path Profile to the next network node. When the Requested Profile reaches the destination node (containing the Client Application requests and network's capabilities mapped into the Path Profile), the I-NAME Server Application answers with a specific message. This message is generated in collaboration with the destination node, indicating its availability to support the Requested Profile. If the destination node answers by an Accepted Profile, it expresses the destination's availability for resources announced in the Requested Profile. Destination node sends a message indicating the acceptance for requested parameters.

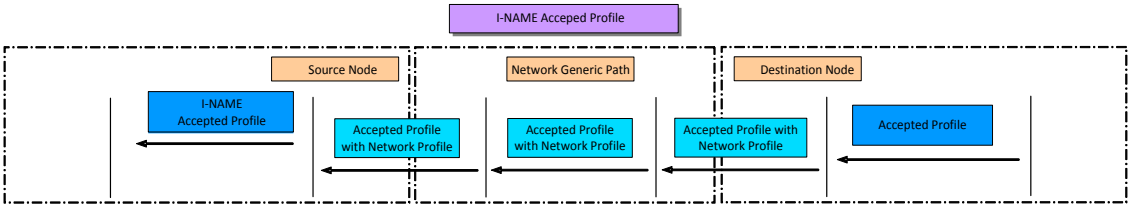


Figure 5-5: I-NAME Accepted Profile message flow

If the destination node answers by an I-NAME Negotiated Profile, it expresses the destination's possibilities for resources announced in the I-NAME Requested Profile. Destination node sends a message indicating the available possibilities for requested parameters.

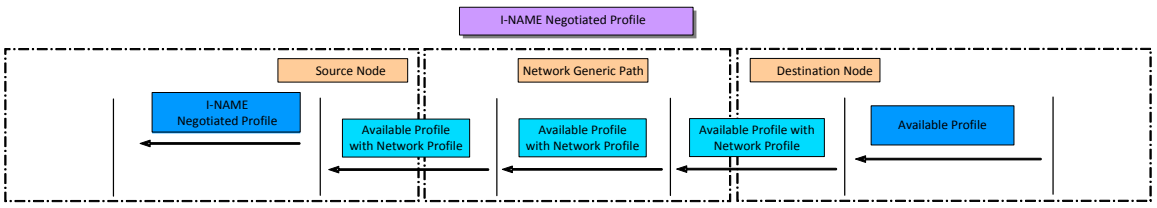


Figure 5-6: I-NAME Negotiated Profile message flow

The answer sent back to the source node records the best path to the destination. Thus I-NAME assists low level routing (based on low level QoS parameter measurements and monitoring), managing in a distributed manner the network resources.

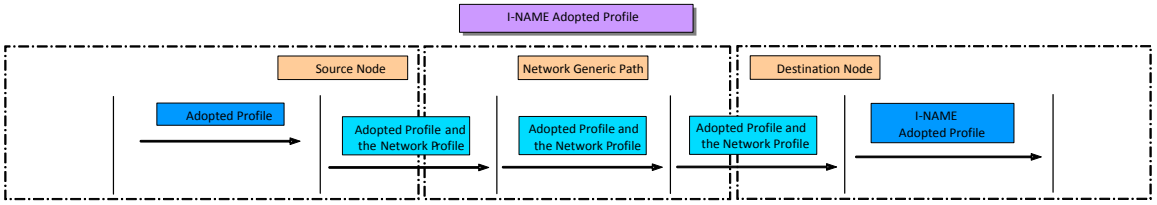


Figure 5-7: I-NAME Adopted Profile message flow

I-NAME Adopted Profile expresses the source's availability for resources announced in the I-NAME negotiated/accepted profile and is generated by the source node.

5.2.2 Resource optimization

Network resources are normally shared. An effective and efficient usage of shared resources requires the establishment of an optimization mechanism or process. Such process actively analyses the variations on the status of the network, providing a way to assist decision and allocate resources. Even when it does not directly act over the network, optimization (as an INM application) provides hints on the best choices to be made, according with the current status of the network, with the prediction of future status of the network and with the context requirements.

Self-adaptation, in the scope of 4WARD, is the capability of each INM node (and therefore of the network), to react to changes insuring the proper network behaviour, according to the defined management policies.

The idea is to include in the INM process the policies of the network that will influence the user's access, and the load balancing in a forecasting basis. Also, one of the novelties is that this is done through INM.

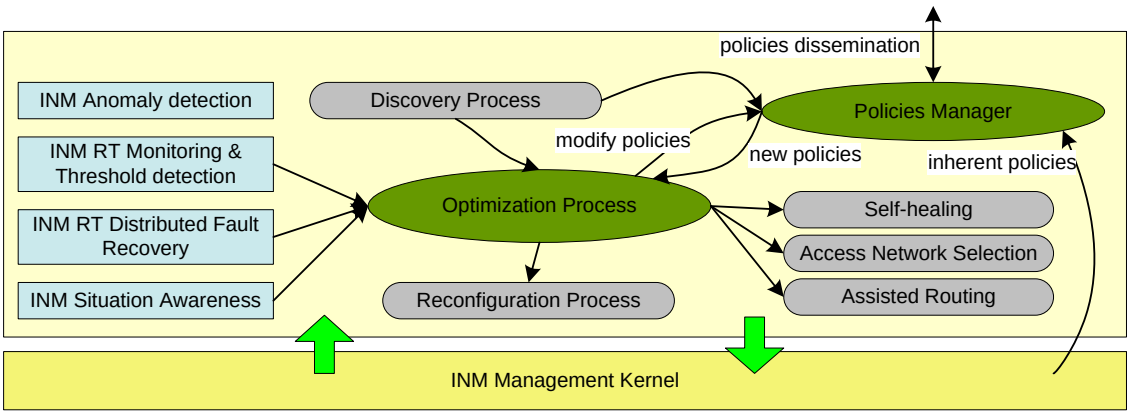


Figure 5-8: Self-adaptation and the optimization process

Figure 5-8: highlights the relations between the optimization processes and other INM monitoring applications (blue rectangles). The grey rectangles represent other self-adaptation functions.

As illustrated, optimization includes the support of local resources optimization (through predictive approach) and reconfiguration, access-network selection and assisted routing (as optimal path selection). These functions require the ability to discover the nodes, perform network bootstrap, and manage entry/leave nodes.

The relation with the Policies Manager is also of key importance. This allows network entities to share the network goals while contributing to their redefinition when they occur, as a consequence of the global adaptation mechanism.

Access-Network Selection is the choice of access technologies (or interface) in multi-homing environments. INM can assist the decision process for multi-access, multi-provider, predictability of service, etc. This functionality interrelates both with INM Situation Awareness and with INM Real-time monitoring and Threshold Detection applications. Network selection deals with cross-layer optimizations, such as medium optimization, choice of access technologies, inter-provider issues and policies, considering forecasting and decision in a long-term basis.

The self-healing functions of a network are related with protection schemes. On a long-term basis, the network continuously adapts avoiding problematic nodes or links before the problems occur. On a short-term basis, the network must react to an already existent problem. The network and resources optimization functionality will interact with self-healing in the sense that it will, for one side, avoid problems in the networks – by predicting them, and for another side, if problems occur, self-healing will trigger network optimization with the new configurations of the network after recovery.

Finally, the interaction with network configuration planning is straightforward, since the optimization of the network resources and interfaces will provide information on the availability of the required resources and interfaces in the network.

Towards In-Network Resource Optimization

The characterization and definition of the optimization mechanisms and related distributed protocols require a better understanding of network nodes interactions. These interactions are based on node characteristics: its type, role and behaviour within the network. A systematic classification of a network node is necessary considering not only its behaviour but also its supported actions (functionalities) and events.

The network entity should be the starting point of the definition of the functional management architecture. The node architecture must provide a detailed characterization of this entity, attending different viewpoints and contexts. Starting on a highly abstract level, we plan to increase detail for each entity as we move down in this top-down approach.

The identification and definition of relevant network information and metrics are also required. These metrics, the weight they have on each configurable parameter and their relevance to other nodes (sharing) establish a base for distributed decision algorithms and collaborative protocols.

Opposing to the previous strategy, a bottom-up approach can be used as a validating method for the conceived models. The definition of a set of progressive scenarios allows the tuning of our model, testing some of the developed concepts and validating some of the exposed options (more details in the annex A). Simulations of each scenario and their corresponding processes will allow us to compare different models, choosing the proper solution for each different situation. With this iterative process, the overall model can be simplified and complexity can be added without loosing the final perspective.

Towards In-Network Optimization Model

Inside a network we can find many different types of resources and parameters. The solution to an optimization problem is neither straight-through nor consensual. Parameters will typically compete to obtain a local optimal point otherwise the results would converge to a common

optimal solution. A network wide optimization process must weight all these parameters and conflicting requirements, achieving a compromise solution.

Whatever criterion is used for optimization, ultimately the allocation of physical resources is involved. We can start by referring each node's internal capabilities such as available memory or processing (CPU) capability. In this case, the optimization problem is (normally) solved internally inside the node. Managing such resources imposes limits on the number of running processes and the size of the local heap of such processes. To support Vnet, the ability of each node to efficiently manage its own physical resources such as CPU and memory is required.

Handling interfaces and links is a wider type of problem. At least two entities need to be involved in this case. If we're using a wireless shared medium, the number of entities will be potentially bigger. In this case, cooperation between such entities is required to solve the problem. By cooperation we mean the exchange of each entity's knowledge on the network, and the iterative and distributed process of decision making.

Although the configuration of the network directly affects the distribution of the available physical resources (such as those referred before), other parameters can normally be "optimized". Typical QoS parameters such as committed bandwidth, propagation delay, jitter and error rate, for instance, depend on the actual physical path taken by each data flow. Other technical parameters could be used as an optimization criterion: energy efficiency, used spectrum, transmission time, message overhead, etc. The use of specific coding techniques (network coding) increases the efficiency of data links and the achieved communication data rates combining (sharing) the transmission of different sources.

The optimizations of non-technical (in a strict sense) parameters take special attention in our research. Each actor in the network (the user, the operator or the service provider) could define a set of preferences. The total communication cost is an important parameter for the user. Equally, avoiding specific paths (for business reasons) could be a major requirement of a network operator. A service provider could require, for instance, a minimum security level for delivering some data, or a minimum apparent quality (e.g. a video stream).

Time takes a special place in this list of "optimizable" parameters. Whatever parameters are to be optimized, the duration of the whole process is relevant as well as the communication delays. Minimization of knowledge dissemination delays and fast decision are main requirements for the distributed algorithm, but minimizing the data flow delay is an important requirement for the obtained solution.

All these parameters can be used to define a set of (active) Network Management Policies. These policies define a range of values for each parameter and a list of conditions that together work towards a desirable operation goal.

The main component of our model is a local optimization knowledge database (or matrix). This matrix encapsulates everything the node has learned from itself and its neighbourhood. This initial model for the Optimization Process can be found in the annex.

Two main types of parameters are identified: metric related parameters and topological description parameters. The first group of parameters hold the configurable properties of the physical resources as well as the parameters, conditions and restrictions that affects them. The second group contain information on the network topology: how many nodes, which nodes, who connects to whom and by where.

Regarding the ownership of the information, knowledge could be local or external, private or shared. Local knowledge is an inherent property of each entity. When the entity boots this knowledge is implicitly obtained. Local private knowledge will not be distributed to other remote entities. External knowledge is obtained from other entities through message exchange as part of the dissemination mechanisms of the distributed management protocols. External knowledge is necessarily not private (shared) on its source.

The scope of knowledge could be partial or global. Partial knowledge is when at least one node does not have that information (yet). Global knowledge means the information has been shared between all participant nodes.

Completeness of knowledge can be classified as partial or complete. Partial information occurs when some data has not been received or was received with errors. We can calculate a probabilistic value for the missing parts.

Finally, common knowledge is when some information is known to all, and all know the information is known to all.

A distributed protocol is a comprehensive set of rules that define the behaviour of each network entity. Each rule sets an action to each possible condition. An action is a “what to do”. A condition is a “when to do”. The previous definition of a distributed protocol could also be seen as the definition of a decision process. The distinction between “distributed protocol” and “distributed algorithm” it’s thin.

Our goal is to achieve an efficient distributed algorithm for real-time improvement of network utilization and performance. Efficiency is measured in terms of time, communication cost and comparison with traditional (centralized) optimization schemes. Each network entity must solve a part of the problem based on a partial view (knowledge) of the network conditions. Collaboration between entities is required to distribute such knowledge and to enforce, confirm or refuse each other decisions. This research is still in progress. We are currently working on the process of collaboration of network entities – which information is more relevant, which is required, how to deal with incomplete information. A general model has being defined and can be found in the annex. Our next step is to use simulation to tune, demonstrate and evaluate the model of the obtained distributed algorithm.

5.2.3 Specific Algorithm: Wireless Multi-Hop Networks

A specific example of INM is the self-adapting functionality of entities in wireless multi-hop networks, such as Wireless Mesh Networks (WMNs), Mobile Ad Hoc Networks (MANETs) or Wireless Sensor Networks (WSNs). Traditionally, networks are composed out of clients and servers with special network entities in between called routers, which are responsible for data forwarding. Unlike to those networks, each node in wireless multi-hop networks also needs to provide routing functionality on its own. That directly implies that the efficiency of the whole network is based on the collaboration of all nodes and thus each network node must be able to adapt its behaviour according to any network changes (e.g. broken routes). However, traffic forwarding in wireless networks is very challenging as there arise much more problems than in fixed networks paper[94][95][96][97][98].

Unfortunately, approaches that have been proposed mostly aim at a certain type of network (e.g. MANET, WMN or WSN) and just try to optimize and adapt their behaviour within this domain. An example for an ongoing research group is the “Routing Over Low power and Lossy networks (ROLL)” [103], which just focuses on a subset of application areas, namely industrial, connected home/building and urban sensor networks. However, the future Internet won’t be composed of multiple clouds that stand alone for themselves, but are fully interconnected. A wireless sensor’s next hop might be another sensor, but it could be also a highly mobile node as well as a TPA (Internet Transit Access Point). What those approaches are missing is the ability to combine and adapt between different types of networks even expecting different kind of traffic characteristics. High mobility, link and network load, resource constraints and even further conditions might directly affect processes such as node discovery or the selection and combination of different routing metrics (e.g. ETX [99], MTM [100], EDR [101], and WCETT [102]). However, an approach that tries to solve this problem by expecting accurate situation information might fail as there is also a difficulty with the inaccuracy of situation awareness in wireless multi-hop networks. Figure 5-9 exemplary shows different types of networks that will lead to different network conditions and thus points out the need to self-adapt according to the given network situation.

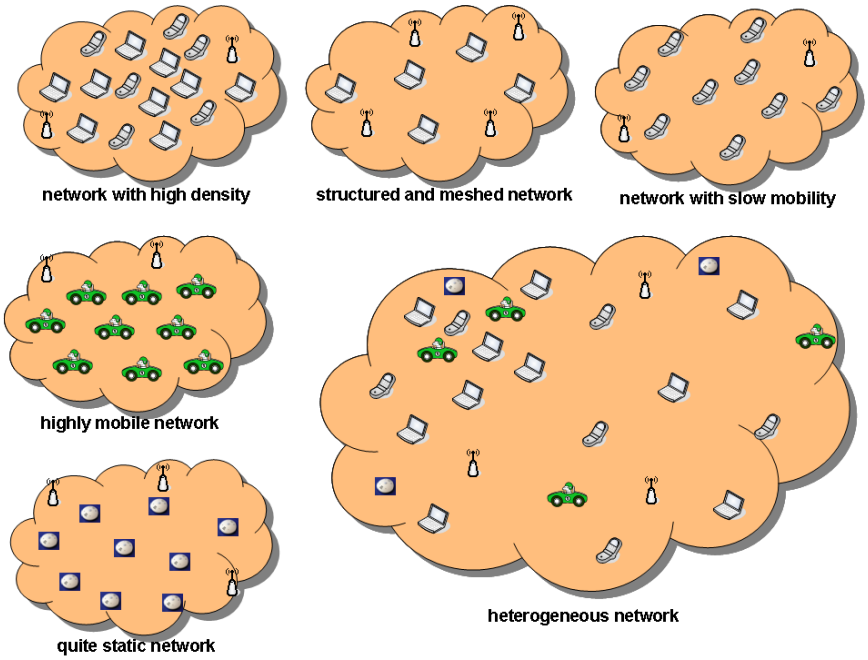


Figure 5-9: Different types of wireless multi-hop networks

The intended INM System acts in a distributed way and allows to efficiently share and balance resources, discover and repair routes on its own, self-adapt to network changes, care about network anomalies and offer a wide range of network state information. Such an INM can be the key enabler to offer a localized and overall picture of network state information to each network node at the same time, depending on the granularity that is requested. Each network entity would know what is going on in the network and thus would be aware about the current network situation of a certain area it is interested in and which might directly affect the decision for the optimization process.

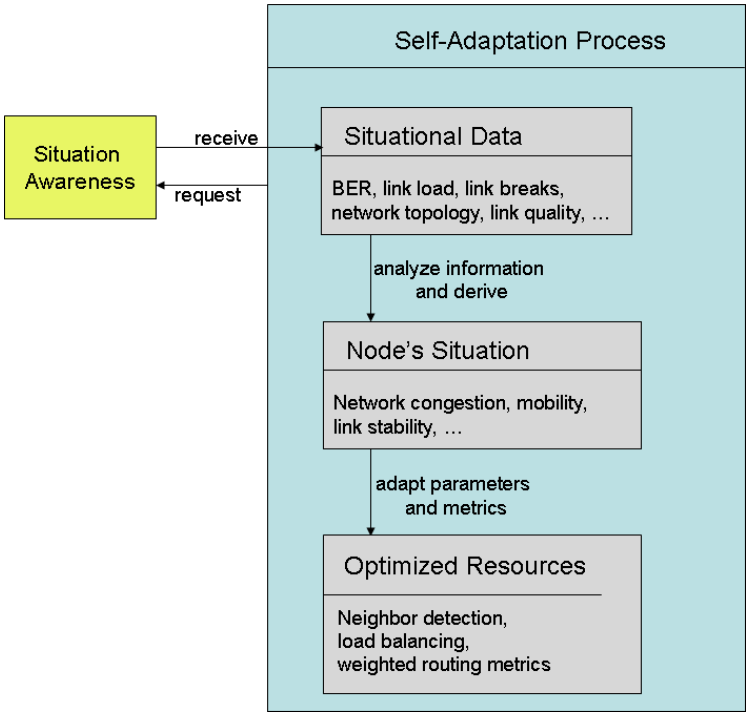


Figure 5-10: Self-adaptation process

The self-adaptation process of a node is exemplarily shown in Figure 5-10. The process requests needed data via the Situation Awareness framework. Those data can be also interpreted as “raw situational information” of a node and include information like network topology, link load or link quality. However, what is really needed to self-adapt to network changes appropriately is the situation or context of a node and not any raw situational data. Hence, this raw situational information will be used as data basis and analyzed by various algorithms and procedures. The outcome of this analysis will be the node’s awareness of its situation which might be reflected e.g. by possible network congestion or link stability.

Following example for mobility shall give an idea how the analysis might look like. Multiple proposals have been made to reflect a node’s mobility pattern and the most appropriate metric seems to be the amount of route breakages as it almost has a linear dependency with the mobility of nodes [105]. Hence, the analysis needs to get information about the link breakages, which can be derived by continuously requesting network topology information from the situation awareness framework and thus realizing any neighbour changes (new neighbours arise whereas old neighbours might disappear). What the analysis then finally will provide is the information about the node’s mobility pattern, e.g. almost static, slowly moving, etc. The main adaptation process will be done afterwards by analyzing the derived situation information and changing appropriate parameters. Appropriate metrics will be adapted for instance to combine and weight different routing metrics (e.g. A (ETX), B (Stability) and C (Less Hop Count)) and thus available network resources can be optimized when a node’s situation changes.

Moreover, the Self-Adaptation Process for resource optimization might also directly influence the topology computation and thus the offered situation awareness information by changing the broadcast interval and broadcast range (number of hops) for neighbour detection or even the routing protocol itself (e.g. from pro-active to reactive approaches). Figure 5-11 exemplarily shows how metrics could be adapted and weighted in case of node mobility, though such adaptation should be not just restricted to node mobility, but at least also include parameters as network load and resource constraints.

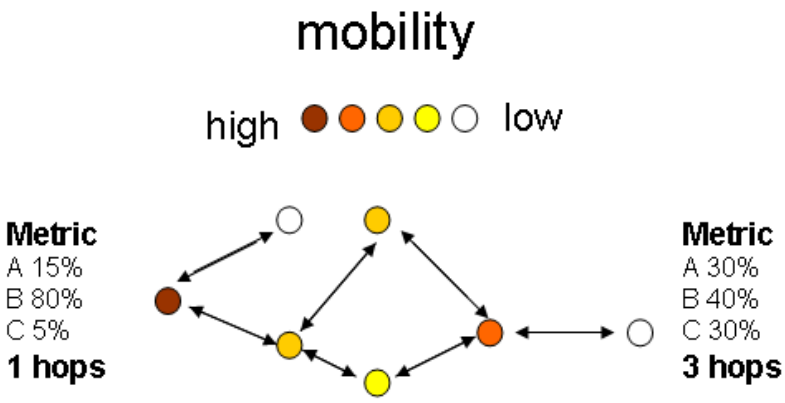


Figure 5-11: Example for metric weighting according against mobility

There is also some work done in WP5 that target towards wireless multi-hop networks. However, the difference between the approaches is that WP5 mainly concentrates on developing new metrics that are interference-aware or deal with grid computing in a wireless multi-hop environment, whereby work done in WP4 mainly targets at the self-adaptation property of network nodes that change their behaviour and even might change used metrics according to a given network situation. It is expected that simulation results will give an impression how metrics are affected when a node’s situation changes, which also might change implicitly by means of situation changes of its neighbouring nodes. Moreover, it is anticipated to get a clear indication about the needed adaptivity for the dissemination of topology information (e.g. in case of mobility or congestion). By continuously adjusting those

parameters we hope to improve the overall route quality, network stability, and scalability and this way increase the whole network throughput.

5.2.4 Network Configuration Planning and Verification Functionality for INM

Traditionally, the planning of activities on an operating network (like configuration changes, upgrades and so on) has been an important part of network management. We believe that INM should include mechanisms that support these operations, based on the intelligence that is pushed inside the network, according to the INM principles.

The aim of the work described in this section is to investigate the possibilities of embedding configuration planning functionality into in-network management. We propose to have mechanisms inside INM that support the pre-planning of network configuration changes before actually performing them. Besides planning of the configuration changes that are going to be committed to the live network, we also propose a functionality to verify the potential effects of these configuration changes.

It is also a goal to investigate how the network administrator (or end users) would, despite the automation, still interact with such a system to perform network administration and planning tasks on a "higher level" compared to today. It is quite likely that operators will still want to have a control terminal in the NOC, displaying the overall network status, and allowing interventions when necessary. This section aims to present a problem statement and a possible scheme for planning network operations in the future INM-enabled networks. The goals for future work are the study of possible implementations and evaluation (feasibility, performance). While it is another important topic, this study does not cover the initial planning of a network.

The self-organizing behaviour proposed for INM has some clear benefits compared to traditional network management. It is probably safe to assume that INM will have a predictable behaviour in reaction to different events in the network, however, the expected behaviour should be known also to the operator or the user, who might not be familiar with the internals of the INM implementation. The expected response of the network to changes should be known before any changes are applied, especially if the desired changes could cause problems or undesired effects. A change in the network could mean anything: connecting one or multiple new nodes, adding/removing links or changing their properties, changes in the network topology or in the nodes, etc. But changes could also be not hardware-related, for example changes in network policies or in resource allocations. Basically, to all the possible actions on the network from the operator/user that INM is designed to handle, it should be possible to predict the expected outcome, since it is the INM mechanisms inside the network that react to the changes.

Fully automatic reconfiguration might not be desirable in all situations: there could be scenarios where due to some changes in the network the self-organizing mechanisms could cause unstable behaviour like load transients or oscillations, even if only for a short transitory period. Moreover, a certain resistance towards fully automated systems can be observed. Operators' fears towards such systems are due to the existing risk that they could lose control over the system. Another valid concern is that bugs can severely cripple such automated systems (even more if they are distributed) The administrator or operator should always have the possibility to control the self-adapting behaviour: it should be possible to change parameters of the INM self-configuration, override some default actions, or even disable some INM functions, if it's needed. In general, we believe that the operator should be able to control the degree of automation / autonomy of an INM-enabled system and should be aware of potential undesired effects with the aid of INM mechanisms.

Network Configuration Planning in the INM Application Space

Network planning is considered to be a separated network management function, according to the classification based on the degree of embedding described in Section 3.3.

From the real-time / non real-time classification point of view, network planning functionality can be imagined in both. In the NRT case, the prediction of the effects of changes would not take into consideration the real-time parameters of ongoing sessions in the network. This is certainly the simpler case. In the second case, real-time parameters and ongoing sessions are also considered, so the effects of planned changes in the network also include effects on the ongoing sessions. Planned, and later, applied changes in the network could also affect ongoing active sessions, not only the overall state of the network. If a planned change could have an undesired effect on ongoing communications (e.g. interruption, delay/jitter outside the agreed QoS limits) this must be signalled in advance. In this case, the task becomes certainly more complicated, we could call this case “Advanced Network Planning”.

Architecture of the Network Planning Application

Figure 5-12 shows a basic architecture for the Configuration Planning Application, and the entities it interacts with.

The Network Planning application will communicate with other entities through the I-NAME / INMP protocol. These entities can be:

- Applications the planning application depends on, running either on the local or on other nodes. Since the communication is handled by I-NAME, therefore the location of the peer applications (on the local node or on remote nodes) is not important.
- Other instances of the Planning Application, running on other nodes.
- External entities, mainly the operator, who is using a management application. The management application supports and uses I-NAME.
- The INM kernel, through the INM Services.

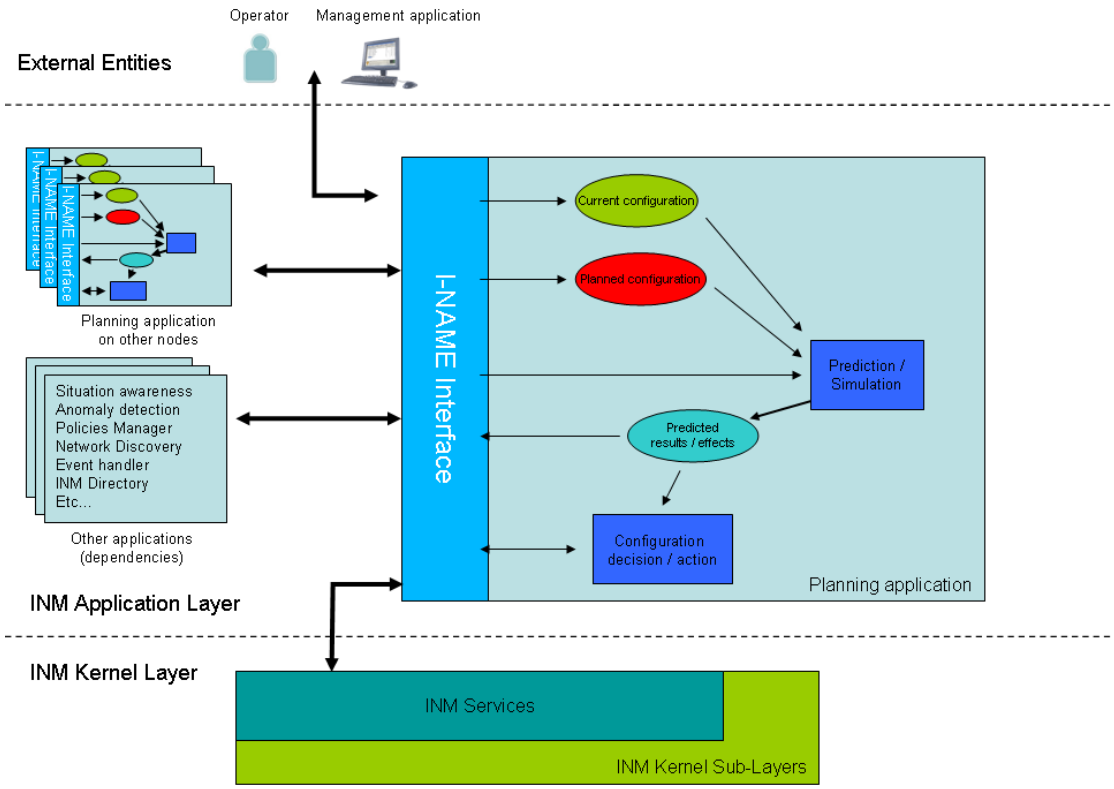


Figure 5-12: Architecture and external interfaces of the Network Configuration Planning application

Network Planning Operations

When changes are needed to an already operating network, the network administrator should be able to plan the changes by interacting with the INM functions through a Management Application. The interaction should include the following steps and will manipulate three main *pieces of information*, as shown in Figure 5-12.

- Analyzing the *current configuration and state of the network*, including its operating parameters, especially any performance problems, anomalies or undesired behaviour. These data should be obtained from the Situation Awareness and Anomaly Detection functions and will be displayed by the Management Application.
- Introduce the *planned configuration* changes through the Management Application. This is not a complete configuration for the network, just changes which are planned to be executed. It does not include state information.
- Receive a response from the INM planning module with the *predicted results* of the changes. The operator can inspect the result using the Management Application. This information includes the network state information which could result after the configuration changes in the planned configuration are applied. These data are provided by the Prediction module, which could employ different methods to produce them: simulation or prediction based on previously learned data.
- Depending on the response, the administrator or operator can choose to apply the changes or repeat the planning process and create a new planned configuration if the results are not acceptable. In case the new configuration is applied, the Management Application will send this command to the Planning Application, which in turn will apply the changes via the INM Services.

These operations can be considered an enhancement of the original NM control cycle with an additional small control and verification loop, as shown in Figure 5-13.

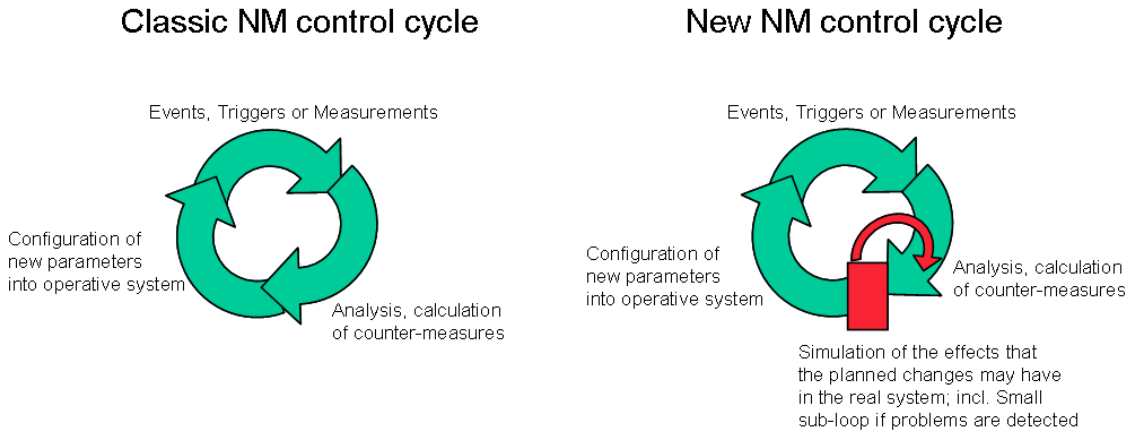


Figure 5-13: Classical and enhanced INM control cycle

5.2.5 Event handling in INM

Automatic configuration of basic functions of the INM entities is an important feature to enable plug-and-play feature in 4WARD. Distribution of events between nodes is one of the basic functions that support other management functions. A typical event can be an exception reporting the anomaly detection described in Section 6.2, a failure in general, or any information which needs to be distributed in an asynchronous manner. In the traditional network management approach, events are generated on the network nodes and reported to a central manager, which normally executes a root cause analysis engine. The relation between monitored nodes and management station in is normally statically configured.

When moving to the INM paradigm, the management logic is not concentrated on a single manager and nodes themselves can process events generated by other nodes, handle

events, trigger self-optimization mechanisms or reporting the problem to a human operator. However, having assumed that a management component can generate an event, a mechanism is needed to define how such event can be forwarded to other nodes. Specifically, it has to be determined which other node can act as a sink of events and has the capability of correlating events or handling them.

We defined a generic framework, on which a complete distribute event handling mechanism can be built, which includes elements with the following roles:

Event generator: it generates events to be sent to another location of the network. Any 4WARD component has the capability of generating events. Referring to the SNMP framework, this role is similar to an agent.

Event handler: it handles events for two purposes: (i) filtering and aggregating events; (ii) correlating events through correlation engines. This role can be taken by any correlation engine presented today.

Optimizer: it receives an event to generate an action. Normally the event received is the result of previous elaborations and is here named Root Cause, to indicate that the cause creating the warnings in the network has been identified. The action taken by the optimizer is related to the specific root cause identified and aims at either optimizing the performance or reacting to a fault in order to heal the network. This role can be taken by any healing component in today's networks.

User Interface: it receives an event and displays it to the administrator.

Algorithms for Event Distribution

Given one or a set of devices acting as generators, we need to deliver the events to the proper location and guarantee that the events are properly handled. We assume the following:

(i) an event generator does not know in advance the proper destination of an event;

(ii) different event handlers can be used for the same event;

(iii) associations between generators and handlers are not unique and stable over time;

(iv) the output and the timing of an event handler is constrained, but unpredictable;

Assumptions (i) and (ii) reflect the distributed nature of the architecture, in which no central manager is present and processing logic is distributed on different nodes. Assumptions (iii) and (iv) imply that an event generator should be able to contact different handlers and that the same handler can produce different results.

Following the initial study on benchmarking of distributed architecture, two main approaches can be followed in order distribute an event in the framework: *sequential* and *parallel*.

In the sequential approach a generator is allowed to select as destination for an event only one of the available handlers. Different criteria can be used to select the best handler given an event of a certain type. The most straightforward way is to select the one for which the network delay is lowest, to achieve timeliness. However, other techniques can be based on past experience, taking into account the success rate measured for previous events. We assume that a handler sends back a reply, according to the result of its processing rules. If a negative reply is received, a generator needs to select another destination. In the worst case a generator needs to try one by one all the handlers. This might be time consuming and therefore not satisfy timeliness requirements.

The parallel approach attempts to reduce the total time required to find the proper handler. A generator can send the event to more than one handler at the same time; duplicate processing can be avoided through the self-organization structures of Section 3.5.3. The maximum number of destinations, n, is a parameter of the algorithm. Clearly, the higher n, the more is probable that one of the chosen handlers is successful. The drawback is that the traffic generated can be very high and the network can be flooded with events.

We introduced a model that aims at treating the correlation engine as a “black box”, meaning that any technology specific correlation engine can be modeled with this black box with certain levels of fidelity.

We assume that the output of the reception of an event can be a success or a failure, considering the ability of generating a root cause as a successful elaboration. The result of a correlation engine is not always fixed, but it depends on the conditions of the network. For our purposes, we are not interested to know *which* root cause would be produced for which event, but only *if* a root cause would be produced. Therefore, we assume that an engine can receive a certain set of events, and there is a certain probability of success upon reception of an event. Further details are available in the paper [93].

Simulations and Results

We have implemented event distribution mechanisms using the OMNeT++ simulator, developing the modules *Event Handler* and *Event Generator*. We set up a reference topology with 6 event handlers with different network delays. We ran simulations with two different sets of values for the probability of success of the event handlers: a homogeneous scenario in which all handlers are characterized by the same probability and a non homogeneous one in which closer handlers are characterized by lower probability.

Table 5-4 shows average results both in the homogeneous and non-homogeneous scenarios. Results are shown for the sequential case and parallel case. In the parallel case, we varied the maximum number of destinations an event can be sent to at the same time, from 2 to 6.

Looking at the distribution of the completion times, as shown in Figure 5-14: for the homogeneous scenario, it can be noticed that in the sequential case times are more distributed towards high values, whereas they shift towards lower values in the parallel cases.

	Homogeneous scenario		Non homogeneous scen.	
mode	completion time (avg), ms	traffic (avg), # messages	completion time (avg), ms	traffic (avg), # messages
sequential	111.1	1.9	417.3	3.4
parallel 2	37.3	6.0	218.9	10.3
parallel 3	28.9	11.1	173.7	18.0
parallel 4	29.7	14.3	171.8	21.8
parallel 5	31.4	16.3	161.9	25.2
parallel 6	29.1	17.9	159.0	26.0

Table 5-4: Completion time and traffic results of simulations with 1000 events generated and computational capability of 10 events

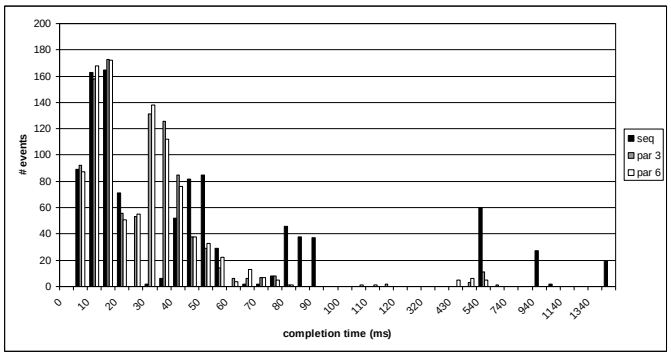


Figure 5-14: Completion time distribution, homogeneous scenario

When looking at the results about completion time and traffic consumption, it is clear that a trade-off exists between the two metrics. The completion time with a parallel approach results much lower with respect to the sequential approach. Sending the event to multiple handlers increases the probability that one of them is able to handle it with success. Increasing the number of destinations can decrease further the completion time. However, the parallel

approach has the evident disadvantage that the traffic generated increases linearly with the number of parallel destinations. The trade-off between timeliness and traffic consumption cannot be resolved, but necessarily the expected behavior of the system should be defined in the governance process of the INM framework. In addition, we propose that nodes support the different distribution mechanisms, and they should be able to switch from one to another on the basis of network conditions or requirements.

In the histogram in Figure 5-14: we notice the presence of high values, approximately between 400ms and 1400ms, especially in the sequential case. These correspond to the occurrences in which all handlers fail in the correlation process or the proper handler is the last in the sequential chain. Such high completion times might not satisfy timeliness requirements that are characteristic of certain events. Therefore, we suggest including inside an event an expiration time, after which the distribution process is stopped and a negative reply is sent to the generator. This would avoid unnecessary traffic and a generator would be informed soon about the impossibility of handling the event and could take further actions accordingly.

Figure 5-15: shows that the limitations of computational capabilities tend to saturate the performances of the system. The sequential mode is more resilient with respect to these constraints, while the parallel mode tends to saturate the system faster. Under very constrained configuration, the disadvantage of consuming more traffic is not compensated anymore by the gain in time. This might not be an issue when considering a small scale network or a single service; nevertheless, when considering performances of a large scale service network, it should be taken in account that a certain planning of the computational resources is required and mapped properly to the parallel distribution mechanism, in order to guarantee performances of the fault management process.

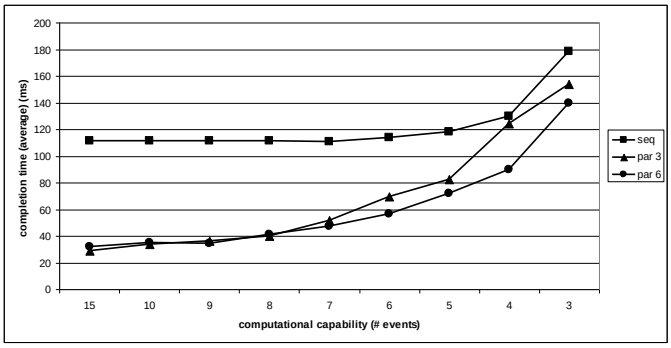


Figure 5-15: Completion time in relation to computational capability, homogeneous scenario

5.2.6 Decentralized Probabilistic Management

This work proposes a probabilistic management paradigm for solving some major challenges of decentralized network management. Specifically, we show how to cope with 1) the overhead of redundant information gathering and processing, 2) the decentralized management in dynamic and unpredictable environments, and 3) the considerable effort required for decentralized coordination of management functions.

To this end, we describe a framework for probabilistic decentralized management in the context of INM. We demonstrate how this framework can be applied to a network of information, a novel clean slate approach towards an information-centric future Internet. We show by means of a simulation study in the area of performance and fault management that we can significantly reduce the effort and resources dedicated to management, while we are able to achieve a sound level of accuracy of the overall network view.

Since management functions are in many cases redundant across the network, we propose to turn them on or off randomly. At one point in time, thus, some functions are turned on, while others are not. The specific way of how the activation of management functions is realized may depend on additional system constraints and performance tradeoffs. If all functions that

are subject to the probabilistic management process are instantiated and resident in the memory of the networked device beforehand, rapid switching between on and off states is possible, which allows fine-granular probabilistic control. It is also conceivable to include the installing and uninstalling of management functions in the randomization process, for instance, if constraints in the transient memory of the networking device apply. While memory resources are conserved, CPU utilization increases and more coarse-grained turning on and off is more advisable. In general, different tradeoffs between the cost of activating and deactivating a function and resource savings are possible. For ease of discussion and without loss of generality, we assume in the following the first of the discussed alternatives.

By control and management functions, we refer to classical functions including, but not limiting to, fault, performance, and configuration management. Note that by “random”, we refer to any type of randomness, including pseudorandom and perfectly random processes. While perfect randomness is difficult to achieve with today’s technology, pseudorandom behavior, implemented by popular random number generators, is absolutely sufficient for our applications.

Figure 5-16 illustrates the basic layout of our probabilistic management framework. A set of management functions is running on the node, interacting with the networking functionality directly or through the node’s database(s) or information store(s). A component called the Randomization Process designates the meta-management entity that takes care of randomization of the management functionality. The randomization process can be influenced through various factors and might be configured from an external entity.

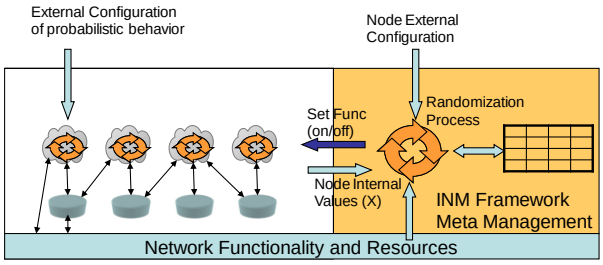


Figure 5-16: Basic probabilistic management framework (node view)

The randomization process on the network element has a probability function fi with a certain probability distribution per management function on the node (cf. Figure 5-16:). An interval Ii denotes the interval between two successive executions of the randomization process for a certain function, which can be fixed, dynamic, or random. Each time the interval Ii elapses, function fi decides whether or not a function is turned on or off. Both probability function and the interval may depend on configuration, type of function as well as configuration and internal information. In a special case, those values might also depend on information about neighboring nodes. However, such models introduce additional, more complicated dependencies again, which we attempt to minimize with the probabilistic scheme in the first place.

Evaluation in the context of Network of Information

We have implemented parts of the INM framework in a Java-based simulation environment and added functionality for creating and running probabilistic decentralized management scenarios. We are able to dynamically add management functionality to a node, which is in turn automatically added to the randomization process handling the respective function. The probability distributions can be parameterized dynamically by setting a specific random number generator in the randomization table.

We apply the probabilistic management framework to a future Internet approach called the Network of Information (NetInf) [4]. Motivated by the fact that users are more interested in the information, rather than the individual nodes storing the information, NetInf defines a new information-centric paradigm. Rather than building on the networking paradigm of node-centric

communication, NetInf exploits information-centricity to connect and relate information elements with one another and to directly build dictionary and management structures on top of these elements. The essence for our purposes is that NetInf provides a global distributed information store, where applications publish information and are able to query information from the system.

In this scenario, we study the effects of probabilistic management in a well-balanced setting in order to understand the effects of the approach in common situations. In the simulations we monitored various values including the number of API requests for retrieving and publishing information, internal cache size, number of transport requests etc. We divided the overall monitoring task into two functions, each of which can be turned on or off randomly. One of the functions monitors the API requests from applications; the other reads data that is gathered in the NetInf system anyway for internal use.

We considered the value of the number of API information retrieval requests. When extrapolating the monitored values to the overall network and monitoring time, we have the same average number of information requests per node independent of the probability of running the monitoring function. The average amount of data gathered per node, however naturally decreases with smaller probabilities (cf. Figure 5-17), therefore less monitoring instances are running in the network. This is as expected in a system with equally distributed activity.

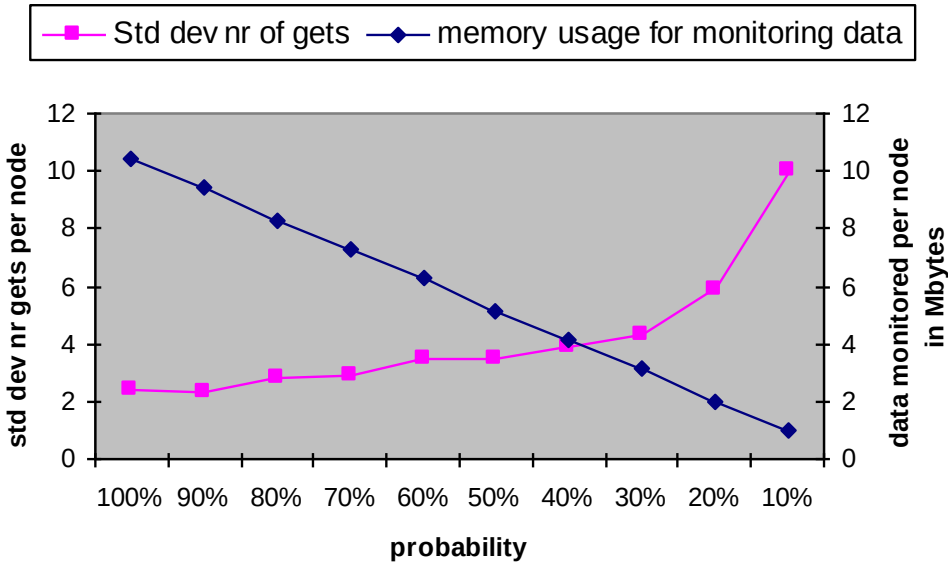


Figure 5-17: Standard deviation and amount of monitoring data gathered.

However, as shown in Figure 5-17, the standard deviation across all simulation runs differs with the probability of running the monitoring function. This means that the extrapolated data's accuracy is smaller than when all monitored data is considered. We observe, however, that down to a probability of just 0.3, which is equivalent to the running of only about one third of the monitoring functions, the standard deviation only changes insignificantly. This underpins the ability of probabilistic methods to achieve accuracy levels that are similar to ideal methods in the scope of network management. Note that the standard deviation is not zero since the there is also variability through the random generation of the network load.

The next set of simulations is concerned with fault management functions. We define a fault as the situation where a node is down and hence, unreachable for responding to object information retrieval requests. We detect a node failure when another node attempts to retrieve an object from the failed node, but was not able to do so. We do not differentiate whether such a failure is due to a network or a node problem. Note that there are several different ways of doing fault management. For simplicity, we restrict the following discussions

to only one possible way that we use to analyze the suitability of the probabilistic management.

For the following simulations we continue to use the previously stated NetInf setting. We randomly choose 100 nodes which at a random time during the simulation fail for a duration of 2 seconds. We have chosen this value because it denotes the boundary where failed nodes are showing up in the form of information retrieval errors. If the failure duration is shorter, there is a set of failed nodes that are not detected, since no information retrieval requests destined for that node fall within the failure window. However, Figure 5-18 shows that also for a probability of 100% to run the fault management functionality, a small number of node failures are still not detected by the NetInf. In order to also detect these failures, targeted fault management would be required that actively checks the nodes' mutual reachability.

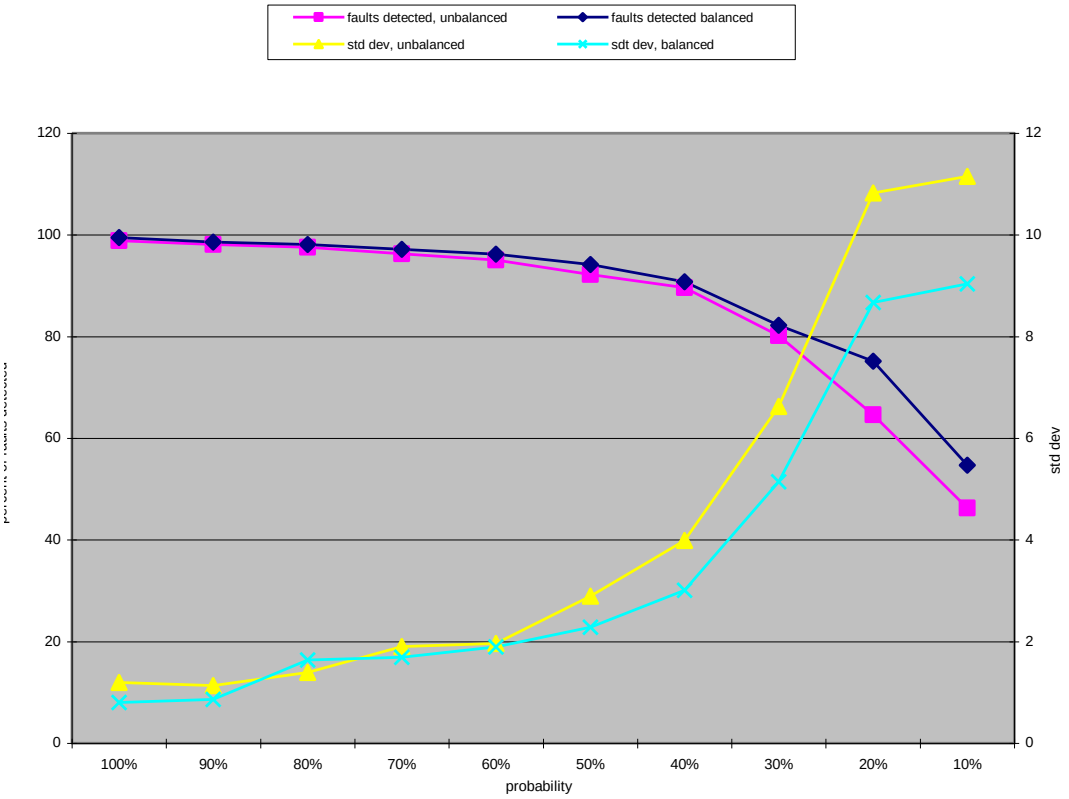


Figure 5-18: Fault detection

Regarding the probability-based detection of faults, Figure 5-18 shows that the fraction of detected faults is fairly small down to a probability of approximately 50%. This observation is true for both the balanced and unbalanced setting, which are both shown in the figure. Furthermore, even at a probability of only 10%, it is still possible to detect about half of the occurring failures. Note that the resource usage decreases linearly with the probability to run a management function.

Comparing the balanced and unbalanced case, the latter shows only a slight decrease in the ability to detect node faults and the standard deviation thereof. This confirms that the probabilistic management paradigm is not only applicable for homogeneous scenarios. It also tolerates inhomogeneous load models and is able to achieve high accuracy.

5.2.7 Congestion Control

A network is considered as congested when too many packets arrive at the same router's queue, resulting in an amount of packets being dropped. To avoid a congestion collapse congestion control is essential. Unfortunately the 'obvious' ways to implement a window-based end-to-end transport protocol as TCP can result in exactly the wrong behavior in response to network congestion. To realise congestion control for future networks in 4Ward we therefore

address the task of congestion control inside the network as an integral part of in-network management. Historically, congestion control was considered to be addressed by the end-system in accordance with the design philosophy of the end-to-end principle. With the dynamics introduced by mobility and the enhanced processing capabilities emerging with next generation networking technologies, these arguments need to undergo substantial revision. Congestion control is a functionality, which should be moved into the network for higher efficiency and better service quality; and local congestion handling becomes an application that can much profit from the INM paradigm.

Local congestion handling is based on control procedures at the routers to prevent, handle and notify congestion. The Internet research differentiates between:

- Local or “dual” congestion control supported by the routers based on collection and evaluation of router traffic information
- End-to-end or “primal” congestion control integrated at the sender to control the sending rate or window size (RFC 2914 [78] ,TFRC [79], RFC 4828 [80], [81], [82], [83], RFC 3742 [84]. The local and end-to-end approaches can be combined dependent on the requirements and infrastructure. In the Internet community, the IRTF Internet Congestion Control Research Group (ICCRG) and IETF Congestion and Pre-congestion notification Group (PCN) are focussed on development and standardisation of congestion control technologies. A comprehensive survey of approaches for congestion control in packet networks is found in [81].

Congestion Control Based on Emergent Behaviour of Pulse-Coupled Oscillators

Majority of congestion control approaches are based on packet dropping strategies. While taking such strategies into consideration, for 4Ward our aim is to go a step further by adding support for redirection of traffic from congested areas in the network. As a prerequisite an interaction between congestion control and routing is required. To reduce management and configuration complexity of a resulting in-network congestion control with a tight coupling to routing we have a strong emphasis on emergent behavior based principles.

An example for emergent behaviour is the spontaneous phase synchronisation of pulse-coupled oscillators, which is a well known phenomena in biology and physics [74][73]. In Figure 5-16 we provide an abstract model of oscillators as introduced by Mirello and Strogatz in [73]. In their model a single oscillator is characterized by a concave status function f(t). For t growing from 0 to T, the function f describes the charging of the oscillator up to a maximum voltage V_{MAX}=f(k*T) (k=1,2,3,…). In case an oscillator is charged up to V_{MAX} it sends a pulse of energy to all oscillators in its environment (i.e. it fires), sets it charging level to 0 and starts again the charging process. At the same time, the energy emitted increases the charging of

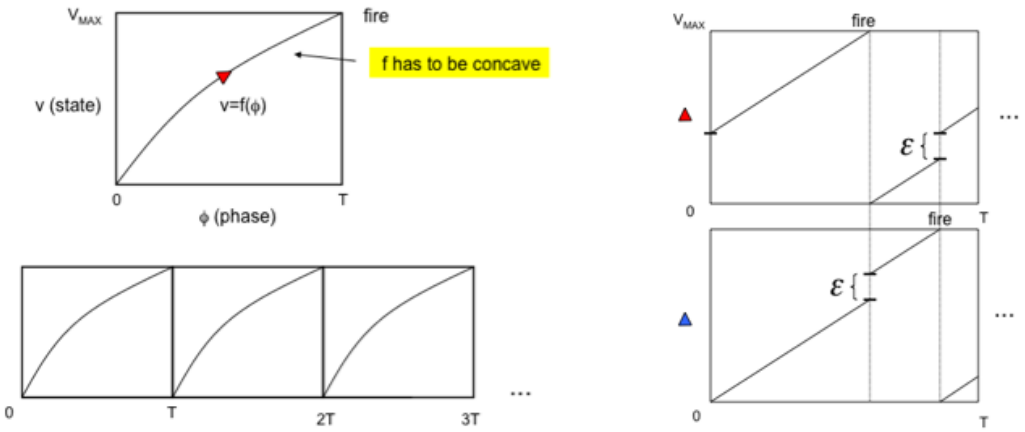


Figure 5-19: Oscillator model of Mirello & Strogatz

each other oscillator by a predefined amount ε, shown in the right side of Figuer 5-16.

A result of the work of Mirello & Strogatz is that each group of identical, pulse-coupled oscillators following their model reaches a state where they have synchronized their phases (up to a set of measure zero). A further property of such oscillators is the fact that in case the frequency of one oscillator is (up to some extend higher) it increases the group frequency to its own **Error! Reference source not found.**. An application of this synchronisation property to networking problems is of interest because of the following reasons:

- The observed synchronisation property is based on emergent behaviour. No additional configuration or management is required.
- Once archived, synchronicity corresponds to a stable equilibrium.

Recent research in the networking area has investigated in the question if pulse coupled oscillators can be used e.g. for time synchronisation in Ad-Hoc networks **Error! Reference source not found.Error! Reference source not found.**. In contrast to this work we focus on the question: “Can pulse-coupled oscillators be applied to congestion control in IP based networks”. To approach this, we follow a two step approach:

Step 1

We start by identifying the filling level of router queues with the frequency of oscillators following the Mirello & Strogatz model as illustrated in Figure 5-20.

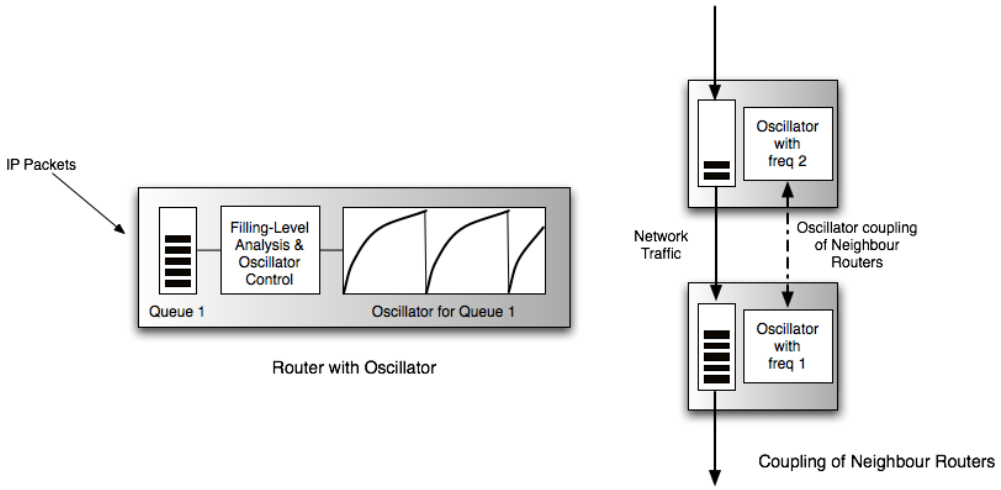


Figure 5-20: Approach and coupling of routers (simplified view)

With a first set of experiments we analyse the effect of network transmission delays in case of groups of such oscillators distributed in the network. Since the network topology used to interconnect oscillators plays an important role, we will further analyse the impact of different network topologies interconnecting the oscillators.

As the result of the experiments we will obtain practical experience with regard to optimal parameter selection for the oscillator internal mechanics as thresholds, charging curves and fire capacity for real networking scenarios. In addition we collect a reference set of topologies to be used to interconnect oscillators as well as information with regard to their properties.

Step 2

In a second set of experiments we evaluate the impact of coupling strength and frequency changes to the synchronisation phenomena.

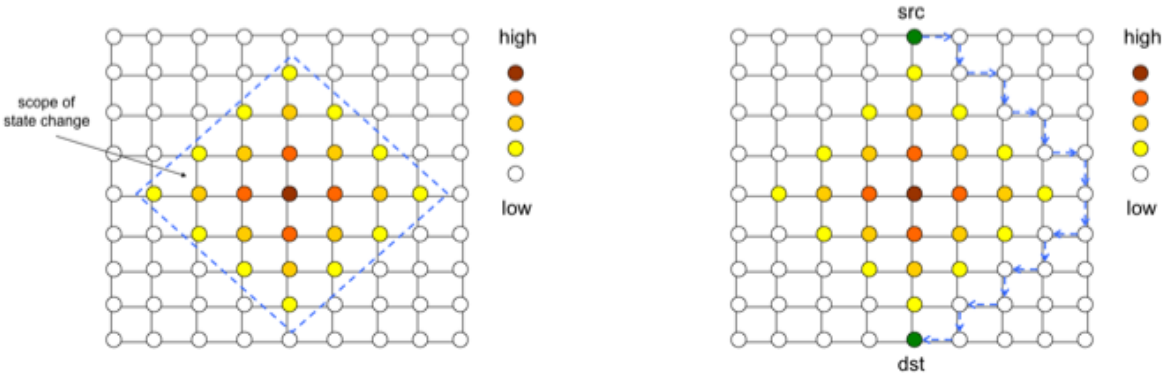


Figure 5-21: Congestion notification based on frequency change

The target is to derive coupling strategies allowing to realise frequency changes with limited scope as shown in Figure 5-21. In case of a high congestion level, the frequency change of one oscillator causes a local frequency change in its direct neighbourhood. When used for congestion notification, the frequency information allows to route around congested network areas as illustrated in the right side of the figure.

5.3 Lessons for INM, Clean Slate

The test cases discussed in Section 5.1 provide us with useful guidelines that are applicable for a clean slate approach.

It is apparent that in order to implement self management, some level of distributed effort is required. In the absence of a "supervisor", network elements have to collectively take responsibility and action for network management.

The extent at which distributed processing is exploited depends on the problem being solved, the network environment, and performance tradeoffs. In some cases, a large group of network elements collaborate, addressing a larger area of the network. In some other, distributed effort is recommended at higher granularity, forming a larger number of smaller groups, each of them self-managed. Furthermore, the formation of such groups is highly dynamics. Nearby network elements might be added to one or more ad-hoc groups as needed.

Performance tradeoffs considered are the following:

- Scalability: a distributed self management arrangement is more scalable than a centralized (or less distributed) one. Higher granularity of self-management, resulted from forming smaller groups of collaborating neighbours, provides a more scalable solution.
- Robustness: As distributed self management is exploited, a larger number of smaller groups of nodes are independently implementing the self management tasks. Clearly, such arrangement is more robust; a failure affects the self management performance of a smaller area of the network, while other collaborating groups are still functional.
- Adaptivity / response time: A more distributed arrangement facilitates a faster self-adaptation. It is easier and faster to detect a property (or threshold), when the number of participating network elements is smaller. Each collaborating group is independent, thereby adapting autonomously to the changing conditions faster than if it were handled by a central management entity.
- Functionality. There is no clear guideline. Depending on the problem being solved, and the network environment, a different level of distributed effort might be preferred. For example, when network-wide throughput needs to be maximized, a more central approach might reach better results, configuring all the elements in a coordinated manner. This, of course, can be achieved only at the expense of other performance tradeoffs. Functionality is sometimes the most important feature, and we might be

ready to pay the price that comes with it. In most cases, however, we want to find an acceptable balance between the expected functionality and the benefits materialized by exploiting distributed effort.

Sometimes, however, better functionality is reached with more distributed effort. In such cases, clearly, we can enjoy all benefits.

A tricky question is which self management functionality lends itself better in a distributed environment. Generally speaking, when the management problem has local scope, it is better handled in a distributed manner (e.g. local re-routing, due to a link failure). If the problem cannot be divided into somewhat-independent regions, then it might require a more centralized approach.

- Overhead. There is no clear guideline. Clearly, the distributed approach entails higher overhead at the setup stage (i.e. more messages and processing effort, forming and adjusting the collaborating groups). However, once distributed, the problem might be solved with less overhead. Our test cases are really the first attempt to implement self-management in a distributed manner. Consequently, the implementation is rudimentary, and there is a lot of room for improvement. We did not gain sufficient experience in this area to form a conclusive guideline.

6 Interworking with other Work-Packages

INM has diverse potential relationships with other work packages. There are five specific relationships that have been considered to be particularly interesting and selected for more detailed analysis and work:

a) How INM architectural constructs relate to the architecture proposed in New APC when specifying the components which will enable the networks of the future.

b) Using in-network management for supporting Generic Path mechanisms. This is described in Section 6.5 on INM instantiation with a focus on QoS and information from hardware layer for GP purposes.

c) Applying self-management to WP6's Network of Information [16].

d) Using WP6's Network of Information mechanisms for INM purposes. This is described as one candidate approach in Section 3.6 on data storage and retrieval. Additionally, the joint Task TC46 is elaborating on this concept.

e) Application of INM to management of Virtual Networks (VNETs).

While (a) and (b) show how WP4 can be useful for other work packages, (c) shows the opposite, i.e. what WP4 may use from another work package (WP6). Thus, both directions for interworking of WP4 with other WPs are considered and described. The interworking with NetInf (WP6) has been investigated more in detail, because a dedicated Task has been set up in the project.

6.1 Architectural Components for the Networks of the Future (WP2)

WP2 is exploring the development of a design process for combining existing, or specifying and generating new networks with customized architectures. WP2 proposes two main architectural constructs which aid them in their task: a Stratum and a Netlet.

A ***Stratum*** is composed by a set of Nodes that are connected through a medium. The stratum encapsulates a set of functionalities that are distributed through the nodes. There are a number of different stratum types defined: governance, knowledge and horizontal.

The basic building block of the Nodes is the ***Netlet*** that provides the basic functional blocks that allows the instantiation of the functionalities inside the Nodes. They can be considered as containers that provide a certain networking service. The Netlet has two main sections, a data related one and a control related one.

Table attempts to map the WP4 architectural constructs to the two WP2 constructs specified in the previous paragraph. The WP4 high level constructs are listed first (Bold) and then any missing WP2 constructs (Bold) are listed and mapped. Figure 15 gives a visualization of the mapping to aid understanding.

WP 4	WP2	Comment
Self Managing Entity	Stratum	The stratum may be one instance or a stratum which contains other strata. These strata are horizontal strata. To keep in mind the governance stratum will cut down vertically through this mapping. The SEs may be atomic of may be a full composition of SEs. This mapping may not be exact but there is a relationship between these two constructs.
Functional Component (with both service and management)	Netlet	Both are containers which reside on a node and handle data (i.e. service) and management functionality.

Management Capability	Functional Block (relative to control)	Management logic.
The logic associated with the service interface inside the FC	**Functional Block (relative to data)**	INM doesn't specify service logic explicitly in the FC, but if the FC has a service interface, it is implied that logic exists to handle this.
Maps to a slice which includes all SEs that expose an organisation interface	**Governance Stratum**	This is the enabler for management objectives to be pushed into the system.
Maps to a slice which includes the MCs which implement the monitoring and situation awareness algorithms	**Knowledge Stratum**	The monitoring and situation awareness algorithms are data sources which provide knowledge for the system.
Dedicated INM Entity	**Control Netlet**	There may be a part mapping here. Both constructs have no link with a service (i.e. data) they are only concerned with management (i.e. control)

Table 6-1: WP4 – WP2 Mapping of Architectural Constructs

6.2 Storing Management Information as Information Objects (WP6)

How can the concept of information objects, consisting of indirection (i.e. naming/addressing mapping or resolution) provided by NetInf, and storage/retrieval be used for decentralized, in-network management? NetInf allows creating more or less persistent information objects via an API, which are then accessible via an indirection and resolution mechanism. For the storage part, two options are possible:

- either the NM or network state information is stored either on the network nodes that produced them, and in that case, NetInf is only used to create a NetInf representation of the real object;
- or, the information object is itself handed over to NetInf, which then in addition to provide the mappings, takes care of the storage task, on nodes that it determines itself.

For retrieval of the information object(s), it is also possible that a only the search of the node where the objects are hosted is part of the NetInf, or that dedicated transport mechanisms are used to get the data object to the requested place in the network.

We have suggested to separate the data handling from the management part and procedures. We will now sub-group the types of data that usually occur in NM. Separation of the overall NM data can help to analyze how and where the concept of information objects can be applied in a beneficial way. The typical control loop involves data that could be seen as separated into at least three groups for our purpose:

(1) control commands, which are many times just modelled as information objects, but could get made available with other mechanisms (not available in the traditional NM protocols);

(2) measurement data about what happens on network elements;

(3) state information, which is fairly local, but could potentially be relevant for others (known also as "triggers", "notifications", "events" or "alerts", depending on further classification).

If we consider that the NetInf API (cf. section 3.6.1) offers two basic types of services that can be used by NM, we can arrange them together with the NM data types in one table in order to see which services apply to which data types, and in which way. The first one is searching for objects according to certain criteria. The second one is storage/retrieval of found or known objects in the sense of "queries", or, at least a "lookup" service. There is a third one, and this is the mapping or resolution of objects IDs onto physical (routable) locations. It is part of the storage/retrieval service but worth to be recognized as a separate service that could be considered a DNS-like service.

		NetInf Methods		
		Search / Retrieve (Query)	**Storage (Object creation)**	**Resolution / Mapping**
NM data types	**Control commands**	Determine nodes to control based on certain criteria	(certain control procedures might be stored as conventional objects)	Determine physical location of one or more target nodes
	Measurement data	Determine nodes to exchange measurements with, or search specific measurements on a set of nodes	Aggregated measurement data	
	State information	Determine nodes to send triggers to and store trigger information within NetInf objects.	Aggregated state information	

Table 6-2: NM data types and NetInf methods

Regarding NM control commands, it is imaginable for nodes that issue control commands to other nodes to use an object ID as an address. The object ID would represent one or more target nodes, and it is the task of the NetInf system to resolve this. There is certainly more flexibility in this approach compared to the relatively simple domain name on IP address mapping that DNS (or even DynDNS) can offer. The resolution process for a NetInf information can potentially be based on many more input parameters.

Mechanism for Mapping of Network State with NetInf

The basic idea of using NetInf for INM network state information can be summarized in the following procedural description:

1) Some state information (subtype “special condition”, i.e. with relevance to other nodes, for example a failure of a hardware component) is emerging at a specific network element (NE) or link between several NE.

2) The affected NE (or another one nearby that detected the failure, too) uses the NetInf API (either locally on the node or at a well-known address in the network) and calls “create object”, selecting a meaningful name or description of the event, for example “[type of failure].[node ID].[time].[geo-position].[network name]”. The NetInf machinery is now able to answer queries from arbitrary NE, looking for matches to this specific state description, NE condition etc.

Since the name of an event is sometimes synonymous with the related information to be transferred (e.g. “failure at node xyz”), the second part, i.e. the data retrieval from some remote nodes can be omitted in these cases. NetInf may store this minimized network state without reference to other nodes.

3a) A network element that is interested in the particular state of another NE would send (via the NetInf upper API) either a “retrieve object” command (if it knows the object ID) already, or a “search” command to get a list of NE that match with the state / condition that the searching NE is interested in. NetInf indirection and resolution search and return all matching entries, for example within a certain geographic area or a managed part of the network.

3b) Variation: Differently from (3a), the “interested” network element can passively subscribe to a certain query, and NetInf would then use a “push” method to inform the interested NE

We can distinguish active and passive retrieval mode. Both may be needed for different types of management applications. The pub-sub qualifies more for *a-priori*-known events and tasks, while the full search makes more sense for the unplanned. The differentiation might apply to the creation of objects in some sense, too, as some state can be “routinely” (e.g. periodically) distributed or published, while other state may become relevant in unplanned ways and times.

Information entities that are conforming to the NetInf approach consist of a binary data object and a separated locator for it. This applies to larger information entities which correspond more to the general NetInf model, where NetInf returns IDs of nodes that hold the requested information.

It is worth noting that there can be a kind of minimal information entity, where the entire information is already fully contained within the identifier, so there is no need for contacting any other nodes for retrieval. It might be useful to support this, depending on the INM use case. Alarm state could use this minimal version (saving time for retrieval via additional nodes), while support of software distribution via NetInf would certainly use the full indirection model and complete NetInf description plus the “binary object” that it refers to.

6.3 Self-Management for the Network of Information

6.3.1 In-Network Management for the Network of Information

Traditionally, network management aspects have been considered in the last stages of designing and developing network technologies. Many initiatives operating under the future Internet umbrella continue to do so, regardless of whether their overall approach is evolutionary or clean slate. Deliverable D-6.1 [4] on the Network of Information (NetInf), reports on the first steps taken towards integrating self-management functionality from the initial design stages of NetInf, an information-centric architecture for the future Internet proposed within 4WARD. Apreliminary set of guidelines on how to apply INM principles to NetInf is proposed. Through several use cases, the advantages of tightly integrating self-management functionality and network awareness with the actual service delivery in NetInf are demonstrated.

6.3.2 Probabilistic Management for the Network of Information

See above for a detailed description for decentralized probabilistic management. Also the results simulated, when applied to the NetInf has been shown. The results show how a certain, specific INM framework functionality (probabilistic management, can be applied to NetInf management functionality, which is built into NetInf. In our case, only within the simulator, this has been achieved.

6.4 INM Operations for Virtual Networks (Vnet)

In general, there are basically two places where INM concepts apply in the context of Virtual Networking. First, the basic infrastructure requires some management, where INM helps to operate the base infrastructure. Second, the network architectures running within a virtual network will have INM capabilities and functions implemented as well. See Figure 6-1. Finally, those capabilities might need to interact in order to achieve cross-layer optimization. Here the INM boundaries and the behaviour at the boundaries concerning inter-operator issue can show its potential, shielding some information but allowing for getting others for the sake of an optimal operation of the overall system.

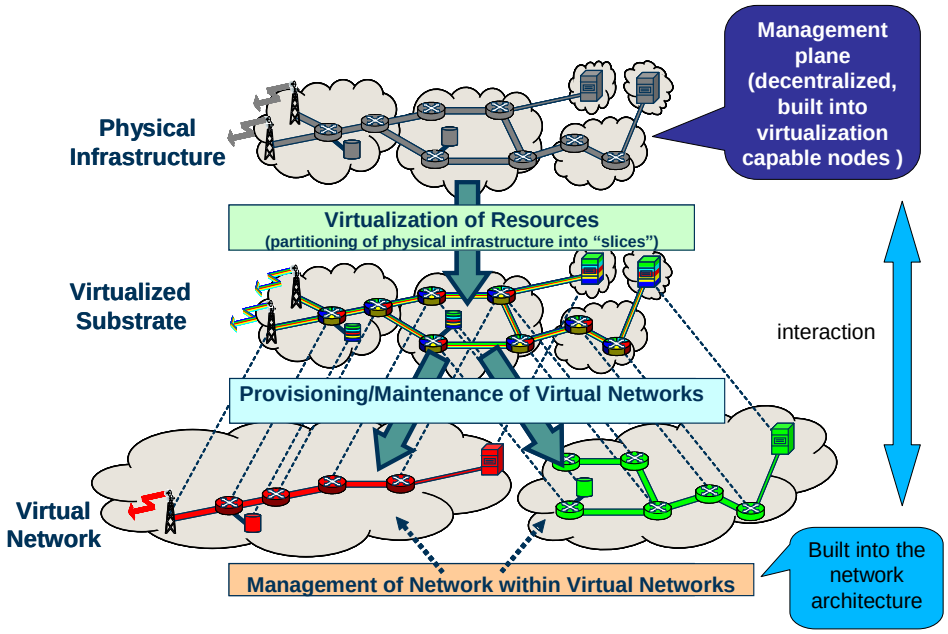


Figure 6-1: Overview of INM in Vnet

6.4.1 Decentralized Vnet Infrastructure Operations

One important challenge on network virtualization is the efficient use of the physical resources. To accomplish such efficient use the management of the physical resources should be transparent to the applications running within the virtual networks, and should be executed at runtime in order to deal with the variation on the load requests of different virtual networks. Traditional resource allocation schemes use offline, centralized, and global view strategies to manage the use of physical resources. In contrast to these strategies, we propose a runtime, distributed, local view approach to manage physical resources.

We define the employment of distribution, local view, and online features on the reallocation of resources of virtual networks. Some assumptions must be observed in order to provide such features in the new scheme, and they are described below.

- The initial deployment of a virtual network is addressed by the provisioning of Vnet architecture described in deliverable D3.1 [4].
- We assume that the virtual topology defined by the first placement will not change during the lifetime of the virtual network, even after the reallocation of virtual slices among physical nodes.
- The reallocation of slices must be as transparent as possible for the virtual node. In the current stage of this research, the reallocation of the virtual slices is transparent in the sense of avoiding exchanging any kind of information between the virtual application inside the moving slice and the virtual managers of the physical nodes involved in the reallocation operation. However, we introduce an interruption time on the execution of the application running inside the moving virtual slice. Interactions in a later stage are foreseen, but complicate the mechanisms, when negotiation needs to take place.

The main objective of maintenance functions is to approximate the virtual node that is generating a great amount of traffic to the destination virtual node. The approximation is done moving the source virtual node from its physical device to another physical device near the destination virtual node

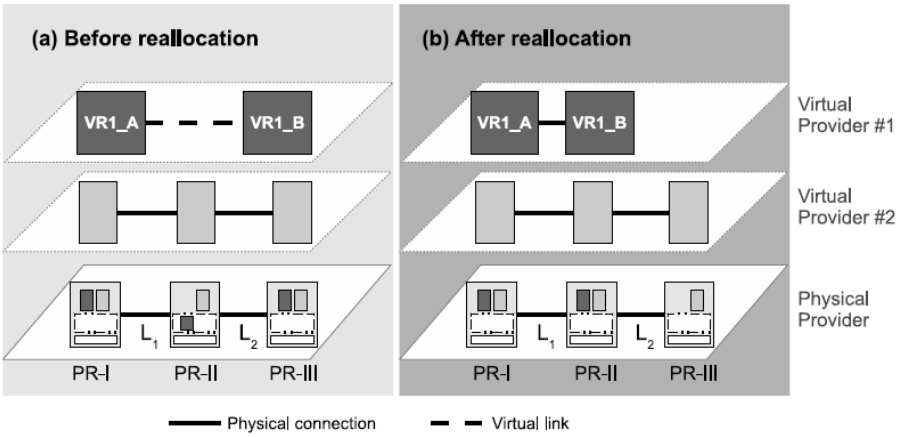


Figure 6-2: Relocation management with INM

6.4.2 Using the Tools of VNET for managing networks

The fundamental tool, which VNET provides, is the separation of functionality into different virtual networks. In the In-Network Management case different management functionality could be separated using this. However, the interaction of the different management functionality is also needed and would need inter-virtual network communication loosening the separation and modularization achieved.

6.5 Management of Generic Paths

4WARD WP5 activities are related to forwarding and multiplexing. The main object of research in WP5 is a new paradigm called the **Generic Path** (GP) [5]. GP is a new communication abstraction. It is expected that due to this new approach the network resource utilisation will be increased. GPs can be created automatically or by explicit signalling. GPs can be of point-to-point type or may have multiple **End Points**. End Points communicate to each other and to the outside world via a binding function and offer one (or at least one) link to participate in networking. Path instances reside in **Compartments** and are interconnected via **Mediation Points (MP)**. Depending on the networking strategy the Mediation Points may operate stateful or stateless, can be dynamic or static, can interpret packets or manipulate packets traversing them. The state awareness varies from null (particular GPs are not recognized, but the effects of MP activities can be observed at End Points of GP, e.g. as with random packet dropping during congestion) to full visibility of GP by the MP (e.g. when resource reservation per flow is done within the network, and a flow being representation of respective at transport level). The compartment defines the scope for addressing and routing (it can be an IPv4 domain, IPv6 domain a LAN or a virtual network, to mention few examples). The Mediation Point located at inter-compartment border can be seen as a multi-layer gateway which can have e.g. data transcoding capabilities, multicast functions, multi-domain gateway functions, etc.

There were no intensive research yet devoted to the management of architecture based on generic paths. Below only initial thoughts are presented. It is envisioned that the management functions will be related to:

- the management of GPs,
- the management of End-Points,
- the management of Mediation-Points,
- the management of Compartments,
- providing of support for GP-oriented operations, e.g. providing information about the network state, etc.

The management of GP should enable the following functions: create, join, leave, destroy, or inspect a generic path. In case when the end-to-end QoS is required, it should be provided by

appropriate resource allocation - it is also necessary to control resource sharing among generic paths inside one compartment in order to provide "fair" and efficient sharing of available resources. It is also necessary to collect resource usage data for accounting.

The management of End Points is responsible for generic and specific end point functions. End points play an important role in GP creation so functions related to GP management should be embedded in End points.

The management of Mediation-Points is used to provide support for required functionality on generic paths and compartments. Management operations have to deal with cross-technology and cross-layer oriented operations. They can be configured in a classical manner and the proper operation of mediation point has to be monitored. Due the important role which mediation point plays in the proposed architecture its operations have to be monitored with a special attention. The mediation point has to participate in resource management and path monitoring.

The management of Compartment should be related to all functions related to paths and compartment creation, maintenance and destruction. Some of those functions can be common to all compartment types while other will be compartment specific. The GP management functions may reside inside a compartment. From the management point of view, in a multi-compartment environment, an important problem is that of resource sharing among compartments. Accordingly, appropriate resource allocation schemes have to been provided. It should be also possible to dynamically allocate or re-allocation the resources in an on-demand style.

It is still to be investigated which management functions will be performed inside compartments and from the in-network management point of view should be treated as a higher-level function, i.e. they are excluded from basic network management operations and which management operations should be performed by INM. Joint Task TC45 created in November 2008 will work on these topics. This work is also expected to address the following:

- network monitoring for routing,
- Generic Resource Management Framework (GRMF),
- usage of generic paths for INM intrinsic needs.

Network monitoring developed in WP4 can provide information about the network state which can be used by routing protocols developed in WP5. Routing protocols should be aware of the intrinsic quality of potential paths as well as current utilization and performance status. In this context the common WP4 and WP5 work is focused on generic routing metrics. They include network level metrics (measured at the link, node and path level) and endpoint level metrics (to monitor the GP formed by the different sections produced by routing).

The GRMF enables a description of resources independently of the network technology which identify resources and components of GPs, specify QoS architecture, evaluate transmission scheduling for GPs and evaluate resource sharing A key functionality within this framework is to identify the resources offered by different network technologies and to describe them in a standardized way. Due to the standard way of description of the offered resources, the GRMF manager is a heterogeneous resource-sharing mechanism. The routing metrics will be also described using the GRMF.

Another common activity of WP4 and WP5 is the usage of generic paths for intrinsic INM information exchange. Such approach is natural but it comes with some problems of creation and maintenance of management-devoted generic paths without INM support (chicken and egg problem).

7 Discussion and Conclusions

7.1 Main Achievements

The main achievements and results of WP4 during the first year of the project can be summarized as follows.

- We demonstrated, through a set of uses cases, the potential of INM capabilities that are not feasible with current management technology. This work is documented in deliverable D4.1.
- We produced the first version of an INM framework design that realizes a management plane inside the network. It supports the embedding of management functions and provides reusable components to build collaborating self-managed entities. This framework is described in Chapter 3.
- We developed a set of algorithms and concepts for real-time management in large networks. An emphasis has been laid on distributed monitoring in large-scale dynamic environments. Extensive simulations have demonstrated the effectiveness of the approach and have quantified the overhead vs. accuracy trade-off. The algorithms can be tuned in real-time to achieve target performance objectives. These results are described in Chapter 4.
- We demonstrated the feasibility of rapid re-configurability for selected management algorithms and functions. We have developed real-time monitoring algorithms that dynamically adapt to node failures and to changes in the network state. Adaptation is performed locally, and, for instance, recovering from node failures can be achieved in a sub-second time interval. This work is described in Chapter 5. This Chapter also describes re-configuration with respect to path restoration and how a fast restoration time can be achieved (at the expense of management overhead).
- We initiated an assessment of potential business value of INM technologies. First results are summarized in Chapter 3.

This deliverable contains additional results, some of which need further evaluation and refinement, which will be performed in the second year.

Based on results from work in this WP to date, thirteen papers have been submitted to international conferences, journals and magazines. Out of those, eight have already been accepted for publication. For instance, four papers will be presented at the upcoming IFIP/IEEE International Symposium on Integrated Network Management (IM 2009), which is the premier venue for publishing research results in network management. In addition, the INM approach has been presented by invitation at several conferences or workshops, for instance on panels (distinguished expert panel at 11th IEEE/IFIP Network Operations and Management Symposium (NOMS 2008), Brazil, April 2008, Information and Communication Technologies Event (ICT Event), November 2008, Lyon) and a keynote at the International Workshop on Distributed Autonomous Network Management Systems (DANMS 08), USA, November 2008.

7.2 Fulfilment of Technical Annex and Measurable Objectives

This section discusses how the technical objectives stated in the Technical Annex and the subsequently defined set of measurable objectives have been approached and to which extent they have been completed or achieved. The technical objectives were addressed as follows:

Technical Objective 1: Evaluate and demonstrate the INM approach to embedded autonomic self-management for selected scenarios.

Task 4.1 has developed four scenarios that illustrate the shortcomings of traditional network management technologies and the potential of INM concepts. The scenarios are described in deliverable D4.1. As originally planned, the INM approach will be evaluated using one or more of these scenarios during the second year.

Technical Objective 2: Provide abstractions and a framework for a self-organizing management plane.

Task 4.2 has defined the basic architectural principles for a self-organizing management plane. It has also developed major elements of an INM framework, for instance so-called management capabilities, which are the building blocks for modeling management algorithms and functions that are developed in Tasks 4.3 and 4.4.

Technical Objective 3: Design and implement a thin pervasive self-organizing network management plane that provides access to and communication between local self-management functions embedded in the network and that organizes itself within a given network and adapts to dynamic changes of network topology and structure. Develop registration and access mechanisms for embedded self-descriptive management functions provided by participating nodes within the management plane.

The current design of an INM framework within Task 4.2 provides the basis for a self-organizing network management plane. This work will be extended during the second year to fully cover all aspects of this objective. Further, the implementation of the management plane (Task 4.5) starts in M16.

Technical Objective 4: Define a scheme, strategies, and protocols for collaborative monitoring, self-optimizing, and self-healing.

Several novel algorithms for distributed real-time monitoring have been developed and evaluated in Task 4.3. Also a distributed algorithm for anomaly detection has been developed and is currently under evaluation in Task 4.3. Lastly, a scheme for in-network self-optimization is under development in Task 4.4. (A potential overlap of this aspect with WP5 has been identified and will be addressed in the remainder of the project.)

Technical Objective 5: Investigate search engine technologies for retrieval of information from an information base that is unstructured, incomplete, timed out and/or faulty.

Mechanisms for information collection and aggregation have been developed. Investigation into search methods with respect to this objective will be performed in the remainder of the project.

Technical Objective 6: Apply the INM approach to virtual networks and a network of information investigated in WP3 and WP6.

A set of guidelines have been defined for providing self-management functionality in the context of NetInf in the joint Task TC46. In addition, work is ongoing on studying distributed storage and retrieval of INM data in the context of NetInf.

The measurable objectives for WP4 were defined and addressed as follows:

Measurable Objective 1: Devise an embedded "default-on" management capability which is an inseparable part of the network itself.

Such a capability is an inherent property of the INM framework design developed in Task 4.2. In the first year, we have developed key components of such a framework. We have defined the basic architectural concepts, including architectural principles and elements that enable the development of embedded management processes and algorithms within a self-organizing management plane. Lastly, we have described how distributed management algorithms that are continuously running in the management plane are integrated into this framework.

Measurable Objective 2: The “default-on” management capability will generate extra value in terms of guaranteed performance in a cost effective way, and will enable the networks to adjust themselves to different sizes, configurations and external conditions.

In the first year, we have provided two specific examples of guaranteed performance in Task 4.3. First, efficient monitoring protocols with performance guarantees with respect to accuracy have been developed. Further, these protocols adapt to changes in network configuration and failures. Second, an anomaly detection protocol with probabilistic performance guarantees has been devised. Task 4.4 has developed several schemes for in-network self-adaptation, including path restoration, resource reservation and resource optimization.

Measurable Objective 3: Such INM capability will allow for a level of real-time awareness of the network behaviour that is not feasible with current management technology.

After the first year we can show that tree-based monitoring protocols can continuously provide the monitored metric with a lag of a sub-second in a realistic network setting with hundreds of nodes (Task 4.3).

Measurable Objective 4: Further, the management plane will typically reconfigure within a sub-second, in response to node addition or failure, and resume correct operation thereafter. Such a capability will significantly reduce the reaction time of today's centralized management systems, specifically in large networks.

In Task 4.3 we have shown that, in the context of real-time monitoring, the adaptation time is very short. For instance, in case of a node failure, our tree-based and gossip-based algorithms re-configure within a fraction of a second.

In the context of event distribution, we have shown that this task is performed within a few hundred milliseconds, enabling fast reconfigurations based on the information conveyed in the event (Task 4.4). In the remainder of the project, we plan to show that the re-configuration of the INM prototype has similar time characteristics.

Measurable Objective 5: For proving cost efficiency, the operational expenses when applying INM will be compared to conventional operation and a substantial cost reduction is expected.

Work is ongoing on assessing the business value of INM technologies. The state of this work is summarized in Chapter 5.

Measurable Objective 6: The time constraints under which management operations can be performed will be evaluated. Since timeliness is achieved at the expenses of computational and traffic overhead, the real-time capabilities will be evaluated with respect to those global objectives.

In Task 4.4, we performed a study that evaluated adaptation time vs. protocol overhead for the purpose of path restoration. This study was performed in the context of studying the effect of decentralization for management tasks.

Measurable Objective 7: To prove scalability of INM, it will be to shown that the number of managed devices or network functions will have a limited impact on the overall performances of management operations, like timeliness.

In the first year, Task 4.3 has evaluated specific algorithms in the context of decentralized real-time monitoring. We have shown, for a given scenario, that the estimation accuracy of our gossip-based algorithm degrades only minimally with the network size, for a given management overhead. Second, we have shown in a simulation study that for our tree-based aggregation algorithm for histograms, the maximum link utilization increases only minimally with the network size.

Measurable Objective 8: Part of the self-discovery capabilities will be verified through practical use cases, where it will be shown that the management plane adds new functionalities as long as they are discovered in the network.

In the first year, Task 4.2 produced the first version of an INM framework design that realizes a management plane inside the network. This self-organizing management plane supports is designed for supporting plug-and-play and dynamic integration or reconfiguration of management functionality, enabling the discovery of new functionality. We plan to provide a practical use case of self-discovery capabilities during the remainder of the project.

7.3 Comparison of 4WARD INM with related projects in EU and US

A range of European and American research projects overlap in scope and approach to some extent with our work in WP4. We list and briefly characterize here some of which we believe are of particular relevance. More information can be found in Annex C.

ANA - Autonomic Network Architecture

EU IP project, 6th FP; Coordinator: ETH Zurich

http://www.ana-project.org/

The focus of this project is on an architecture for autonomic networking that facilitates self-* features. One goal of this project is to include dynamic, adaptive and programmable monitoring features as an integral part of the network architecture. Though both ANA and 4WARD aim at increasing the level of network automation, ANA should be regarded as a generic architecture while INM in 4WARD aims at leveraging a tight coupling of management functions with the services deployed on a device.

E3

EU IP project, 7th FP; Coordinator: Alcatel

https://ict-e3.eu/

E3 aims at a gradual, non-disruptive evolution of current wireless systems into an integrated, scalable and efficiently managed Beyond-3G cognitive radio system framework. The use of the radio resources and spectrum are optimized following cognitive radio and cognitive network paradigms characterized by reconfigurable, self-adaptive, autonomic and collaborative features. Both E3 and 4WARD aim at designing autonomic management systems, with E3 focusing on wireless scenarios, while 4WARD having a wider scope in terms of considered scenarios.

AutoI - The Autonomic Internet

EU project, 7th FP; Coordinator: Hitachi Europe

http://www.ist-autoi.eu/

This project proposes to transition from a service agnostic Internet to a service-aware network where resources are managed by applying autonomic principles. The approach to network management in AutoI is catered for in one of the 5 planes: virtualization, orchestration, management, service enabler and knowledge planes. Similar to 4WARD, this project aims at developing a distributed self-managing management plane. However, unlike 4WARD, this project focuses on virtual resources.

Simple Economic Management Approaches of Overlay Traffic in Heterogeneous Internet Topologies

EU STREP project, 7th FP; Coordinator: University of Zurich

http://www.smoothit.org/

This project looks into the detailed economic and technical mechanisms for a flexible, secure, and scalable traffic management of overlay networks in tomorrow's ISPs. Its objectives include efficient structuring of overlays, advanced traffic management and study of key requirements for commercial applications. 4WARD is related to this project in a number of ways. Overlay networks represent concrete scenarios within the more abstract and generalized virtualization framework of 4WARD. Traffic management is also addressed in this project using similar, distributed and self-organizing concepts as the broader INM approach of 4WARD.

Architectural Support for Network Trouble-Shooting

NSF FIND project; Coordinator: ICSI

http://www.nets-find.net/Funded/ArchtSupportNet.php

This project focuses on troubleshooting in networked systems, including failure detection, identification, root-cause analysis and attributing problems to responsible parties. The approach is based on annotating network traffic with meta-information to facilitate causality tracking of events. Although this project and 4WARD have the common goal of achieving efficient troubleshooting, 4WARD does not consider annotating network traffic for troubleshooting, but instead focuses on algorithms that do not necessarily need such data.

A Framework for Manageability in Future Routing Systems

NSF FIND project; Coordinator: University of Pennsylvania

http://www.nets-find.net/Funded/Framework.php

The goal of this project is the development of a framework for specifying, understanding, and evaluating what features to put into routing systems to make them manageable and evaluate design choices and trade-offs thereof in terms of performance and manageability. The project is expected to deliver results identifying key manageability features that must be incorporated into future routing architectures. The vision pursued by this project is in line with that of 4WARD and we intend to closely study the approach used by this project and their results.

Design for Manageability in the Next Generation Internet

NSF FIND project; Coordinator: University of Wisconsin-Madison

http://www.nets-find.net/Funded/Manageability.php

The objective of this project is to explore design alternatives in the development of technology that will facilitate critical network management tasks in a NGI. The project focuses on low-level building blocks that can be composed in different configurations to enable an Internet management plane. Similar to 4WARD, this project aims at designing automated and intrinsic management solutions. However, the goals of both projects differ in that this project puts more emphasis on test-bed experimentation methodology.

Model-based diagnosis in the Knowledge Plane

NSF FIND project; Coordinator: MIT

http://www.nets-find.net/Funded/ModelBased.php

The goal of this project is to develop a Knowledge Plane (KP), a component of a Future Internet that locates, collects, and manages existing knowledge in a distributed way. This partially overlaps with the goal of Task 4.3 in 4WARD. We therefore intend to closely study the results achieved within this project.

A Network-Wide Hashing Infrastructure for Measurement and Monitoring

NSF FIND project; Coordinator: Harvard University

http://www.nets-find.net/Funded/NetworkWide.php

The focus is on monitoring in large-scale networks. The goal is to reduce monitoring overhead by summarizing monitored data through hash functions. This project is complementary to our current work in 4WARD as this project focuses on aggregation of monitoring data over time rather than over space as we do.

Towards Complexity-Oblivious Network Management

NSF FIND project; Coordinator: Cornell University

http://www.nets-find.net/Funded/TowardsComplexity.php

This project focuses on the development and evaluation of a network management architecture that aims at reducing the complexity of network management. The basic approach is to separate the management plane composed of distributed network devices from the data plane, and, requiring the data plane to expose a generic management interface to the management plane. This project has goals similar to 4WARD and proposes similar approaches. We intend to closely study the results achieved within this project.

7.4 Plans for the Second Project Year

At the current state, the design of architectural elements and their interaction including basic mechanisms has been completed. Work for the second year includes:

- Specifying architectural details, such as the modelling and detailing of interactions and more complex management processes, in conjunction with prototypical implementations.
- Showing by means of prototypical implementations how the framework supports scalability and how external conditions can be propagated into the self-organizing management plane and to the management capabilities of individual management algorithms.
- Investigating in conjunction with a prototypical implementation how the framework supports algorithms and management processes in achieving real-time and performance requirements.

The plans for the definition of monitoring functions include:

- Analyzing monitoring protocols under highly dynamic scenarios. For instance, under high node and link failure rates.
- Designing algorithms and mechanisms for the secure estimation of global aggregates in multi-domain network environments.
- Performing a comparative evaluation of gossip-based vs. tree-based monitoring algorithms.
- Evaluating the distributed anomaly detection algorithm designed during the first year.
- Engineering a design for continuous estimation of traffic matrices in a distributed fashion.

The plans for the definition of self-adaptation functions include:

- Completing the research started during the first year. The concepts will be further investigated, solutions will be provided and implemented, and simulation results will evaluate the validity of the proposals
- Investigating the concept of self-organization, the formation and management of collaboration groups that implement specific INM tasks.
- Compiling results and recommendations, which will be included in the final WP4 deliverable.

Additionally, WP4 will work towards the evaluation and implementation of the concepts developed so far. A prototype that will serve as basis to demonstrate the feasibility of the INM paradigm as well as to show the advanced capabilities defined within WP4. The evaluation study will support the benefits introduced by the INM functions and will be based on D-4.1. Among other things, this working item will include further investigation on the impact of INM on business values for the future Internet.

Finally, WP4 will strengthen the synergies with other WPs and apply results of WP4 to enabling advanced management capabilities in the context of Virtualization, Generic Paths, and Network of Information. The discussions with WP5 showed important opportunity for additional collaboration between INM and the Generic Path on new forwarding and multiplexing paradigms since these new paradigms naturally include a measure of self-adaptation. This is true in particular for routing in both wired and wireless networks where dynamic path choice both improves efficiency and ensures application QoS requirements are satisfied. In the second year, synergies in the respective approaches of WP4 and WP5 will be more efficiently exploited in the newly created joint Task TC45.

References

General

[1] R. Roth (editor), “4WARD Deliverable D4.1: Definition of scenarios and use cases”, FP7-ICT-2007-1-216041-4WARD / D-4.1.

[2] K. Wuenstel (editor), “4WARD Deliverable D-1.1: First Project-wide Assessment on Non-technical Drivers”, FP7-ICT-2007-1-216041-4WARD / D-1.1.

[3] Michael Soellner (editor), “4WARD Deliverable D-2.1: Technical Requirements”, FP7-ICT-2007-1-216041-4WARD / D-2.1.

[4] S. Baucke and C. Görg (editors), “4WARD Deliverable D-3.1: Virtualisation Approach: Concept”, FP7-ICT-2007-1-216041-4WARD / D-3.1.

[5] T. Biermann (editor), “4WARD Deliverable D-5.2: Description of Generic Path Mechanism based on Cooperation, Coding, and Routing Techniques”, FP7-ICT-2007-1-216041-4WARD/D-5.2.0.

[6] B. Ohlman (editor), “4WARD Deliverable D-6.1: First NetInf architecture description”, FP7-ICT-2007-1-216041-4WARD / D-6.1.

[7] C. Foley et al., “A Framework for In-Network Management in Heterogeneous Future Communication Networks”, MACE 2008, Samos Island, Greece, September 22-26, 2008.

[8] C. Dannewitz, K. Pentikousis, R. Rembarz, É. Renault, O. Strandberg, and J. Ubillos, “Scenarios and Research Issues for a NoI", MobiMedia 2008, Oulu, Finland, 7-9 July 2008.

[9] J. Schoenwaelder, “Protocol-Independent Data Modeling: Lessons Learned from the SMIng Project”, IEEE Communication Magazine, Vol. 46, No. 5, May 2008.

[10] J. Strassner, “Directory Enabled Networks”, Indianapolis: Macmillan Technical Publishing, 1999.

[11] C. Adam and R. Stadler, "Patterns for Routing and Self-Stabilization", 9th IEEE/IFIP NOMS, Seoul, Korea, April 2004.

[12] Y. Yemini, G. Goldszmidt, and S. Yemini “Network Management by Delegation”, Integrated Management (IM) 1991.

[13] D. Levi and J. Schoenwaelder, “Definitions of Managed Objects for the Delegation of Management Scripts”, RFC 3165.

[14] R. Carroll, C. Fahy, E. Lehtihet, S. van der Meer, N. Georgalas, and D. Cleary. "Applying the P2P paradigm to management of large-scale distributed networks using a Model Driven Approach", 10th IEEE/IFIP Network Operations and Management Symposium (NOMS), Vancouver, Canada, 2006.

[15] R. Mortier, R. Isaacs, and P. Barham, “Anemone: using end-systems as a rich network management platform”, ACM SIGCOMM Mining Network Data workshop (MINENET), August 2005.

[16] K. Pentikousis, C. Meirosu, A. Miron, and M. Brunner, “Self-management for a Network of Information”, International Workshop on the Network of the Future, Dresden, Germany, June 2009.

[17] A. Pras, B.-J. van Beijnum, and R. Sprenkels, “Introduction to TMN,“ University of Twente, Enschede, The Netherlands, CTIT Technical Report 99-09, Apr. 1999.

[18] F. Sestinim, “Situated and Autonomic Communication – an EC FET European Initiative” ACM SIGCOMM Communication Review, Vol. 36, No. 2, pp.17-20, April 2006.

[19] C. Jelger, C. Tschudin, S. Schmid and G. Leduc, “Basic Abstractions for an Autonomic Network Architecture”, 1st IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC), Helsinki, Finland, June 2007.

[20] Autonomic Network Architecture (ANA). Project funded by the European Union Information Society Technologies Framework Programme 6 (EU IST FP6), http://www.ana-project.org/

[21] D. L. Tennenhouse and D. J. Wetherall, “Towards an active network architecture,” Computer Communication Review, Vol. 26, pp. 5-18, 1996.

[22] A. A. Lazar, “Programming telecommunication networks,” IEEE Network, Vol. 11, pp.8-18, 1997.

[23] G. Pavlou, “On the evolution of management approaches, framework and protocols: A historical perspective”, Journal of Network and Systems Management, Vol. 15, pp. 425-445, 2007.

[24] ITU, “Maintenance: Telecommunication Management Network --TMN Management Functions,” Recommendation M.3400, 1992.

[25] A. Clemm, “Network Management Fundamentals,” Cisco Press, 2006.

Chapter 2 - Motivation for In-Network Management and Scope of Work

[26] 3rd Generation Partnership Project, Technical Specification Group TSG RAN, Evolved universal terrestrial radio access network (E-UTRAN): Self-configuring and -optimizing network use cases and solutions, 3GPP TR 32.816, Release 8 (2008)

[27] A. Ferreiro et al., “Migration guidelines with economic assessment and new business opportunities generated by NOBEL phase 2”, IST IP NOBEL phase 2 document, deliverable D 2.4, 2008 www.ist-nobel.org/Nobel2/imatges/D2.4_final%20version.pdf.

[28] G. Haßlinger, S. Schnitter, and M. Franzke, “The Efficiency of Traffic Engineering with Regard to Failure Resilience”, Springer Telecommunication Systems, Vol. 29, No. 2, pp. 109-130, 2005.

[29] M. Lähteenoja and B. Olsen (Editors), “Techno-economics of integrated communication systems and services: OPEX models”, Celtic project ECOSYS, Deliverable 6 (2005) www.celtic-ecosys.org.

[30] NGMN Alliance, Informative list on SON use cases, Annex A of use cases related to self organizing networks; Overall Description, Version 1.53, Ed. F. Lehser, Next Generation Mobile Networks Ltd. (2008) 1-35 www.ngmn.org/uploads/media/Ngnm_Use_Cases_Self_Organising_Network_2_02.pdf.

[31] T. Rokkas, D. Katsianis, D. Varoutas, and T. Sphicopoulos, “Fixed mobile convergence for an integrated operator: A techno-economic study”, IEEE Symposium on personal, indoor and mobile radio communications, 2007.

[32] TeleManagement Forum, Enhanced Telecom Operations Map® (eTOM) The business process framework for the information and communications services industry, GB921 Vers. 4.0 (2004) <www.tmforum.org> corresponding ITU-T standard M.3050.

[33] S. Verbrugge et al., “Operational expenditures for telecom operators”, 9th IEEE Conference on optical network design and modeling, 2005.

[34] S. Verbrugge et al., “Methodology and input availability parameters for calculating OpEx and CapEx costs for realistic network scenarios”, Journal of optical networking, Vol. 5, No. 6, pp. 509-520, 2006.

[35] S. Wallin and V. Leijon, “Rethinking network management solutions”, IEEE IT Pro, Nov/Dec 2006.

[36] J-P. Martin-Flatin, S. Znaty, and J-P. Hubaux, “A Survey of Distributed Enterprise Network and Systems Management Paradigms”, Journal of Network and System Management, Vol. 7, No. 1, pp. 9-26, 1999.

Chapter 3 - Framework for In-Network Management

[37] D. L. Parnas, “On the criteria to be used in decomposing systems into modules”, Communications of the ACM, Vol. 15, No. 12, pp. 1053-1058, December 1972.

[38] J.C. Wileden, and A. Kaplan, “Software interoperability: principles and practice”, International Conference on Software Engineering, May 1999.

[39] J. O. Kephart, D. M. Chess, “The Vision of Autonomic Computing”, IEEE Computer Magazine, pp. 41-50, January 2003.

[40] A. Greenberg et al. , “A Clean Slate 4D Approach to Network Control and Management”, ACM SIGCOMM Computer Communication Review, Vol. 35, No. 5, October 2005.

[41] B. Ahlgren, L. Eggert, B. Ohlman, and A. Schieder, “Ambient Networks: Bridging Heterogeneous Network Domains”, 16th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, Vol. 2, pp. 937-941, September 2005.

[42] B. Jennings, S. van der Meer, S. Balasubramaniam, D. Botvich, M. Ó Foghlú, and W. Donnelly, “Towards Autonomic Management of Communications Networks”, IEEE Communications Magazine, Vol. 45, No. 10, pp. 112-121, October 2007.

[43] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, “A knowledge plane for the Internet,” ACM SIGCOMM, 2003.

Chapter 4 - Distributed Algorithms for Real-time Monitoring, Anomaly Detection and Situation Awareness

[44] M. Jelasity, A. Montresor, and O. Babaoglu, "Gossip-based aggregation in large dynamic networks," ACM Transactions on Computer Systems, Vol. 23, Issue 3, pp. 219-252, August 2005.

[45] A. Demers, D. Green, C. Hauser, W. Irish, and J. Larson, "Epidemic algorithms for replicated database maintenance," 6th Annual ACM Symposium on Principles of Distributed Computing, Vancouver, British Columbia, Canada, August 10 - 12, 1987.

[46] C. Tang, and C. Ward, "GoCast: Gossip-Enhanced Overlay Multicast for Fast and Dependable Group Communication," International Conference on Dependable Systems and Networks (DSN), Yokohama, Japan, June 28 - July 1, 2005.

[47] A. Ghodsi, S. El-Ansary, S. Krishnamurthy, and S. Haridi, "A Self-stabilizing Network Size Estimation Gossip Algorithm for Peer-to-Peer Systems," SICS Technical Report T2005:16, 2005.

[48] F. Wuhib, M. Dam, R. Stadler, and A. Clemm, "Robust Monitoring of Network-wide Aggregates through Gossiping," Integrated Management (IM) 2007.

[49] M. Dam and R. Stadler, "A generic protocol for network state aggregation," Radiovetenskap och Kommunikation (RVK), Linköping, Sweden, June 14-16, 2005.

[50] S. Dolev, A. Israeli, and S. Moran. "Self-stabilization of dynamic systems assuming only read/write atomicity," 9th Annual ACM Symposium on Principles of Distributed Computing (PODC), Quebec City, Quebec, Canada, August 22 - 24, 1990.

[51] K. S. Lim and R. Stadler. "SIMPSON — a SIMple Pattern Simulator fOr Networks", http://www.s3.kth.se/lcn/software/simpson.shtml, February 2007.

[52] University of Twente - Traffic Measurement Data Repository, http://m2c-a.cs.utwente.nl/repository/, August 2006.

[53] F. Wuhib and R. Stadler, "Adaptive Real-time Monitoring in Mobile Wireless Networks", Technical Report, Royal Institute of Technology (KTH), January 2008.

[54] F. Wuhib, M. Dam, and R. Stadler, "Decentralized detection of global threshold crossings using aggregation trees", Springer Computer Networks, Vol. 52, No. 9, pp. 1745-1761, June 2008, DOI= http://dx.doi.org/10.1016/j.comnet.2008.02.015.

[55] F. Wuhib, M. Dam, and R. Stadler,"Gossiping for Detection of Threshold Crossings", 11th IFIP/IEEE International Symposium on Integrated Network Management (IM), New York, USA, June 2009.

[56] G. Cormode, M. Garofalakis, S. Muthukrishnan, and R. Rastogi, “Holistic aggregates in a networked world: Distributed tracking of approximate quantiles”, ACM SIGGMOD, 2005.

[57] A. Deligiannakis, Y. Kotidis, and N. Roussopoulos, ”Hierarchical In-network data aggregation with quality guarantees”, EDBT, Crete, 2004.

[58] D. Jurca and R. Stadler, “Computing histograms of local variables for real time monitoring using aggregation trees”, 11th IFIP/IEEE International Symposium on Integrated Network Management (IM), New York, USA, June 2009.

[59] A. Gonzalez Prieto and R.Stadler "A-GAP: An Adaptive Protocol for Continuous Network Monitoring with Accuracy Objectives", IEEE Transactions on Network and Service Management (TNSM), Vol. 4, No. 1, June 2007.

[60] R. Baumann, S. Heimlicher, M. Strasser, and A. Weibel, “A Survey on Routing Metrics”, TIK Report 262, ETH Zurich, February 10, 2007.

[61] A. Lakhina, M. Crovella, and C. Diot, "Characterization of Network-Wide Anomalies in Traffic Flows", 4th ACM SIGCOMM Conference on Internet Measurement, 2004.

[62] P. Barford, J. Kline, D. Plonka, and A. Ron, "A signal analysis of network traffic anomalies", ACM SIGCOMM Workshop on Internet Measurement, 2002.

[63] M. Mahoney, "Network Traffic Anomaly Detection Based on Packet Bytes", ACM Symposium on Applied Computing, 2003.

[64] H. Hajji, "Statistical Analysis of Network Traffic for Adaptive Faults Detection", IEEE Transactions on Neural Networks, Vol. 16, No. 5, pp. 1053-1063, 2005.

[65] L. Huang, X. L. Nguyen, M. Garofalakis, M. I. Jordan, A. Joseph, and N. Taft, “In-Network PCA and Anomaly Detection”, Advances in Neural Information Processing Systems 19, MIT Press, pp. 617-624, 2007.

[66] I. Rish et al., "Adaptive Diagnosis in Distributed Systems", IEEE Transactions on Neural Networks, Vol. 16, No. 5, pp. 1088-1109, September 2005.

[67] A. Subbiah and D. M. Blough, "Distributed Diagnosis in Dynamic Fault Environments" IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 5, pp. 453 – 467, 2004.

[68] S. Rangarajan, A. T. Dahbura, and E. A. Ziegler, "A distributed system-level diagnosis algorithm for arbitrary network topologies", IEEE Transactions on Computers, Vol. 44, No. 2, pp. 312 – 334, 1995.

[69] http://www.tmforum.org/

[70] A. Gonzalez Prieto, "Adaptive Real-time Monitoring for Large-scale Networked Systems", PhD. Thesis, KTH, Royal Institute of Technology, November 2008.

Chapter 5 – INM Self-Adaptation

[71] R. Cohen and G. Nakibly, "Maximizing Restorable Throughput in MPLS Networks", IEEE Infocom 2008 mini conference, April 2008.

[72] C. Elster and D. Raz, "Covering All Bases with a Short Blanket: A Dynamic Monitoring Resource Allocation Scheme", IEEE Workshop on Automated Network Management, Apr. 2008.

[73] R. E. Mirollo and S. H. Strogatz, "Synchronization of pulse-coupled biological oscillators", SIAM Journal on Applied Mathematics, Vol. 50, No. 6, pp. 1645-1662, December 1990.

[74] C.S. Peskin, "Mathematical Aspects of Heart Physiology", in "Principles of Electrophysiology with Special Reference to the Heart", Courant Institute of Mathematical Sciences, New York University, New York, USA, pp. 268-278, 1975.

[75] A. Tyrrell, G. Auer, C. Bettstetter, "Firefly Synchronization in Ad Hoc Networks", 2nd MiNEMA workshop, Leuven, Belgium, 2006.

[76] H. Yao-Win, A. Scaglione, "A scalable synchronization protocol for large scale sensor networks and its applications," IEEE Journal on Selected Areas in Communications, Vol.23, No.5, pp. 1085-1099, 2005.

[77] G.B. Ermentrout, J. Rinzel, "Beyond a pacemaker's entrainment limit: phase walk-through", American Journal of Physiology - Regulatory, Integrative and Comparative Physiology, Vol. 246, No. 1, pp. 102-106, 1984.

[78] S. Floyd, "Congestion control principles", RFC 2914, 2000.

[79] M. Handley, S. Floyd, J. Padhye, and J. Widmer, "TCP Friendly Rate Control (TFRC)", RFC 3448, Network Working Group, January 2003.

[80] S. Floyd and E. Kohler, "TCP Friendly Rate Control (TFRC): the Small-Packet (SP) Variant", RFC 4828, Experimental, April 2007.

[81] J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control," IEEE Network, Vol.15, No.3, pp.28-37, 2001.

[82] L. Rizzo, "PGMCC: A TCP-friendly single-rate multicast congestion control scheme," ACM SIGCOMM, 2000.

[83] Ch. Bouras, A. Gkamas, and G. Kioumourtzis, "Smooth Multicast Congestion Control for Adaptive Multimedia Transmission", Next Generation Internet Networks (NGI), 2008.

[84] S. Floyd, "Limited Slow-Start for TCP with Large Congestion Windows", RFC 3742, March 2004.

[85] B. Briscoe, "Flow Rate Fairness: Dismantling a Religion", ACM SIGCOMM Computer Communication Review, Vol. 37, No. 2, pp. 63-74, April 2007.

[86] C. Perkins, E.Royer, and S. Das, "Ad hoc On-demand Distance Vector (AODV) Routing", RFC 3561.

[87] D. Johnson, D. Maltz, and Y-C. Hu, "The Dynamic Source Routing (DSR) Protocol for Mobile Ad Hoc Networks for IPv4", RFC 4728.

[88] C-C. Chiang, H-K. Wu, W. Liu, and M. Gerla, "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel", IEEE Singapore International Conference on Networks, (SICON) Singapore, 16-17 April 1997.

[89] C. E. Perkins and P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance Vector (DSDV) for Mobile Computers", SIGCOMM Conference on Communications Architectures, Protocols and Applications, Aug 1994.

[90] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Laouiti, L. Viennot, and T. Clausen, “Optimized Link State Routing Protocol (OLSR)”, RFC 3626.

[91] B. Bellur, R. G. Ogier, and F. L. Templin, “Topology Dissemination Based on Reverse-Path Forwarding (TBRPF)”, RFC 3684, February 2004.

[92] J. Moy, “OSPF version 2”, IETF RFC 2328, April, 1998.

[93] C. Mingardi, G. Nunzi, D. Dudkowski, and M. Brunner, “Event Handling in Clean-Slate Future Internet Management”, accepted as poster at IEEE IM 2009.

[94] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris, “Capacity of Ad Hoc Wireless Networks”, MobiCom 2001.

[95] E. M. Royer, P. M. Melliar-Smith, and L. E. Moser, “An Analysis of the Optimum Node Density for Ad hoc Mobile Networks”, IEEE International Conference on Communications 2001.

[96] Y. Yang, J. Wang, and R. Kravets, "Designing Routing Metrics for Mesh Networks", IEEE Workshop on Wireless Mesh Networks (WiMesh), 2005.

[97] L. Iannone, R. Khalili, K. Salamatian, and S. Fdida, "Cross-Layer Routing in Wireless Mesh Networks", 1st International Symposium in Wireless Communication Systems (ISWCS), 2004.

[98] H. Lundgren, E. Nordström, and C. Tschudin, “Coping with Communication Gray Zones in IEEE 802.11b based Ad hoc Networks“, WoWMoM, 2002.

[99] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A High-Throughput Path Metric for Multi-Hop Wireless Routing", MOBICOM, 2003.

[100] B. Awerbuch, D. Holmer, and H. Rubens, "The Medium Time Metric: High Throughput Route Selection in Multi-rate Ad Hoc Wireless Networks", WONS Conference, 2004.

[101] J. C. Park and S. K. Kasera, "Expected Data Rate: An Accurate High-Throughput Path Metric For Multi-Hop Wireless Routing", IEEE Sensor and Ad Hoc Communications and Networks (SECON), 2005.

[102] R. Draves, J. Padhye, and B. Zill, “Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks”, MobiCom, 2004.

[103] http://www.ietf.org/html.charters/roll-charter.html

[104] M. R. Endsley, “Toward a theory of situation awareness in dynamic systems”, Human Factors, Vol. 37, No. 1, pp. 32-64, 1995.

[105] L. Qin and T. Kunz, “Survey on Mobile Ad Hoc Networking Routing Protocols and Cross-Layer Design”, Technical Report SC4-04-14, August 2004.

Annexes

A QoS And Resource Optimization

This annex presents more detailed description about QoS and resource optimization.

A.1 *QoSdmFC* Cross-Layer Messages and Primitives (details from 3.7.2)

A.1.1 QoSdmFC: Messages and Primitives using SDL

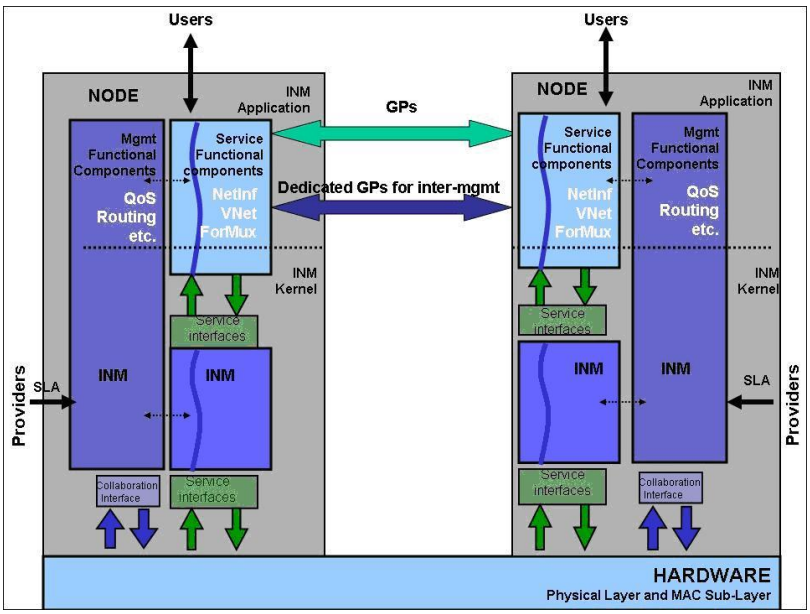


Figure A-1: Framework instantiations based on functional components

This example refers to QoS dedicated management functional component dmFC only, which will interact with other self-managing functional components smFC such as: NetInf, ForMux, as well as with hardware (see Figure A-1). The specification description language SDL involved uses the following symbols:

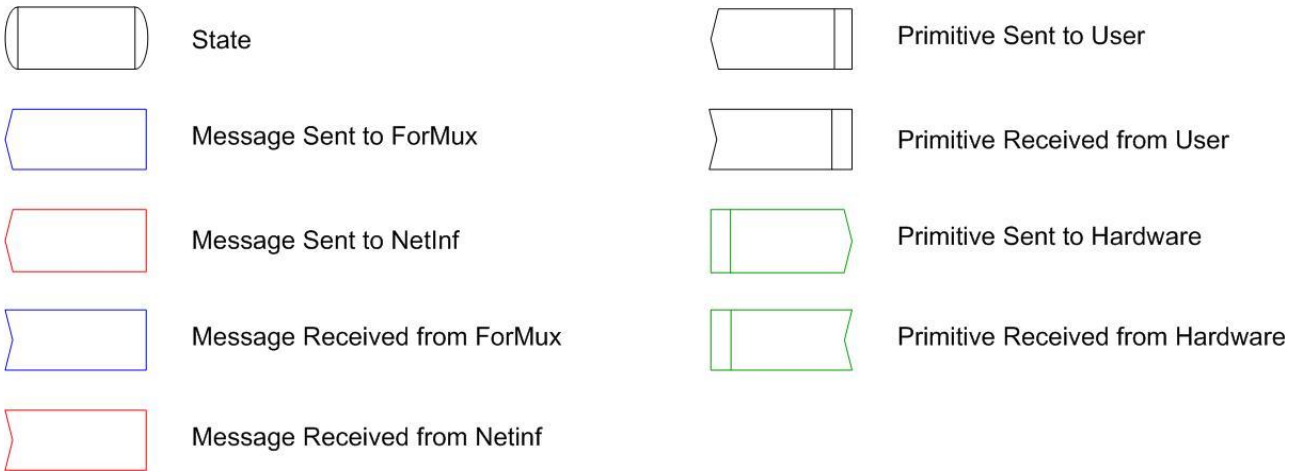


Figure A-2: QoSdmFC: messages and primitives using SDL

Note that management messages are exchanged between nodes through a dedicated management GP (horizontal communication), whilst the primitives are mainly used to

exchange information between users and layers, or in the cross-layering processes (vertical communication).

The QoSdmFC is initially inactive, being in state S0 (Figure A-3).

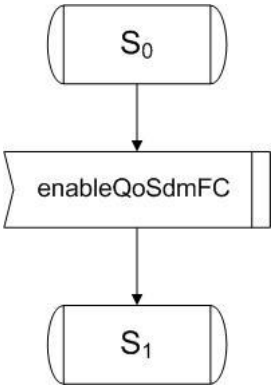


Figure A-3: QoSdmFC, state S0: inactive

Once instantiated, the system will enter in state S1 waiting for top-down cross-layer request (i.e. the committed QoS parameter, usually the transfer rate imposed to the hardware) (Figure A-4). Note that in any of the following states the functional component could be disabled when the primitive disableQoSdmFC is issued. In state S2 if the hardware is not reacting in due time imposed by Refuse Timer, the committedQoS message received from ForMux (Figure A-4) will not be sent anymore as committedQoS primitive (Figure A-5). In case of a successful exchange, the top-down approach ends and the system goes into state S3.

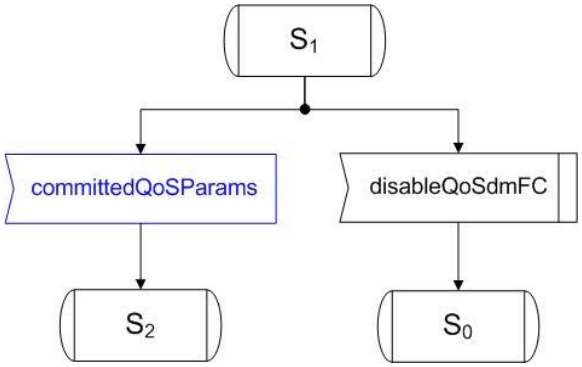


Figure A-4: QoSdmFC, state S1: waiting for top-down cross-layer request

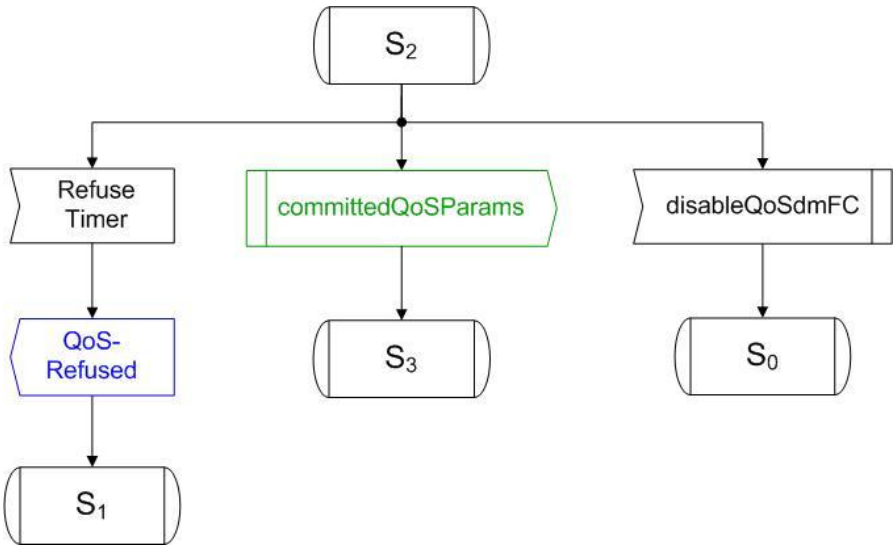


Figure A-5: QoSdmFC, state S2: top-down cross-layer

The bottom-up approach is considered a success if the primitive requestQoSParams is understood by the hardware (Figure A-6). As a result obtained from the hardware, the current traffic parameters BER, BER probability and nominal transfer rate for that specific physical link are included into getQoSParams primitive (Figure A-7).

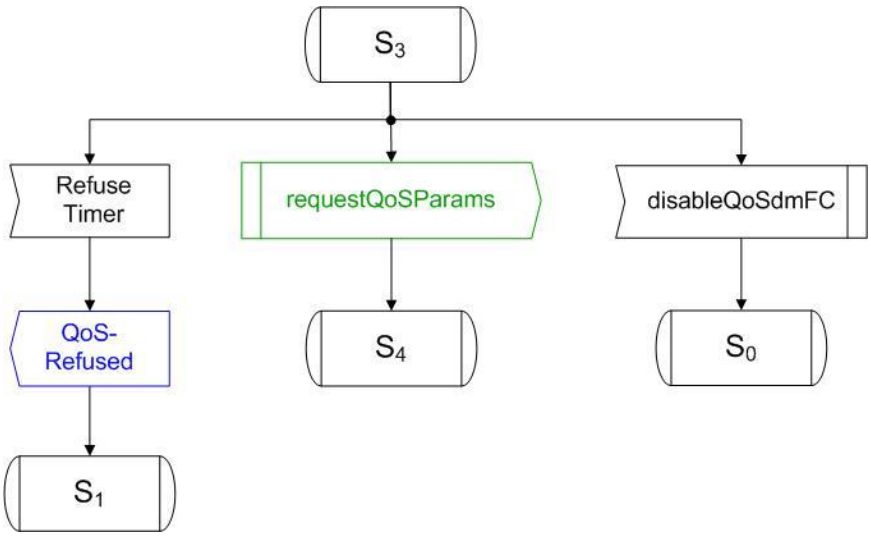


Figure A-6: QoSdmFC, state S3: bottom-up cross-layer request

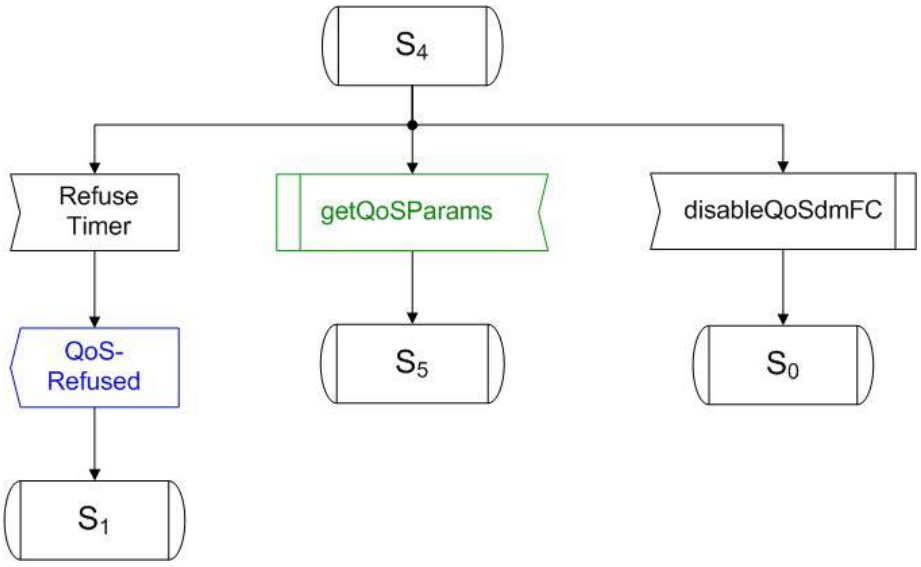


Figure A-7: QoSdmFC, state S4: bottom-up cross-layer

The traffic parameters are converted by INM into a management information object, being published as message publishQoSObject through the service interface to NetInf (Figure A-8). The same information is processed also to calculate a composite metric needed by the INM routing module for global routing table that will assist ForMux and other smFCs in performing routing (Figure A-9). However this global routing table is important also for management purposes, in order to permit/deny any operational access to hardware and to perform routing in case of emergency.

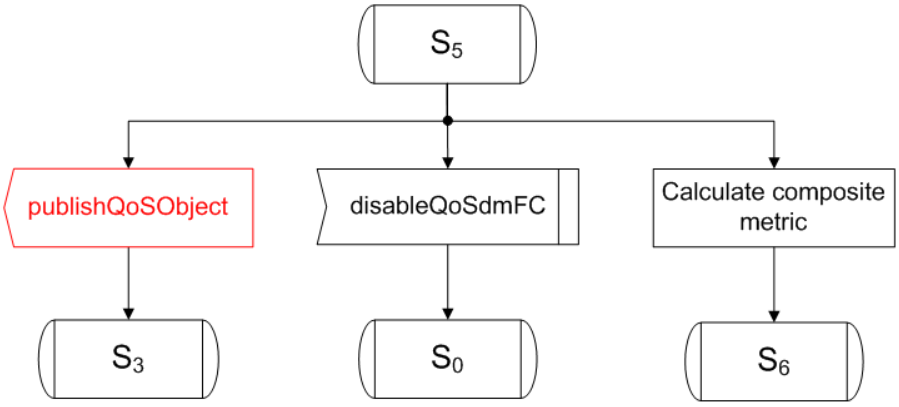


Figure A-8: QoSdmFC, state S5: QoS parameters published as a service

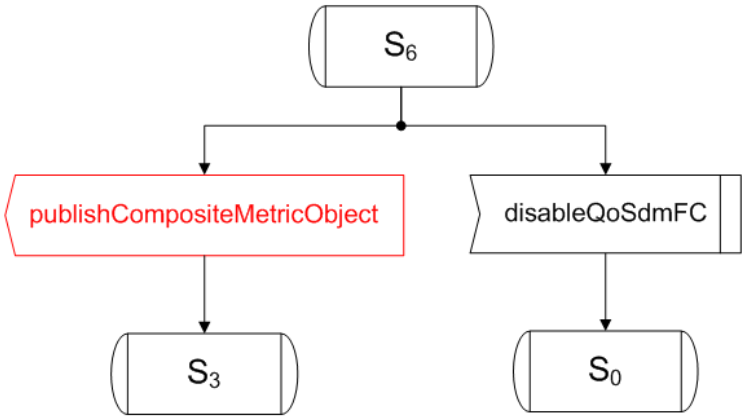


Figure A-9: QoSdmFC, state S6: composite metric published as a service

A.1.2 Proposed XML Format

The messages and primitives described in this paragraph are used by QoSdmFC and are according to the requirements presented in 3.7.2 and in Annex A.1.1. As INM Protocol we kept for the initial phase of implementation the existing IP, whilst for locator we used the existing MAC address. Obviously the XML format permits to replace them once the networks evolve towards Future Internet and other protocols will be involved.

Message *committedQoSParams*

```
<message from="source_name“>
        <type>committedQoSParams </type>
        <node>
                <locator type="Type”> inmp://192.168.1.1/</locator>
                <name>FC/dmFC Name</name>
                <type>Service/Management</type>
                <capability>
                        <locator type="Type”> interface_identifier</locator>
                        <name>Capability_name</name>
                        <interface>Collaboration/Organization</name>
                </capability>
```

```
        </node>
        <parameter committed_value="VALUE" measurement_unit="VALUE" > Parameter_1</parameter>
        .....
        <parameter committed_value="VALUE" measurement_unit="VALUE" > Parameter_N</parameter>
</message>
```

Example:

```
<message from="ForMux">
        <type>committedQoSParams </type>
        <node>
                <locator type="absolute"> inmp://192.168.1.1/</locator>
                <name>ForMux</name>
                <type>Service</type>
                <capability>
                        <locator type="relative"> 00ae.4050.4350</locator>
                        <name>Cross-Layer</name>
                        <interface>Collaboration</name>
                </capability>
        </node>
        <parameter committed_value="2" measurement_unit="Mbps" >Transfer_Rate</parameter>
</message>
```

Primitive *committedQoSParams*

```
 <primitive to="destination_name">
        <type>committedQoSParams </type>
        <node>
                <locator type="Type"> inmp://192.168.1.1/</locator>
                <name>FC/dmFC Name</name>
                <type>Service/Management</type>
                <capability>
                        <locator type="Type"> interface_identifier</locator>
                        <name>Capability_name</name>
                        <interface>Collaboration/Organization</name>
                </capability>
        </node>
        <parameter committed_value="VALUE" measurement_unit="VALUE" > Parameter_1</parameter>
        .....
        <parameter committed_value="VALUE" measurement_unit="VALUE" > Parameter_N</parameter>
</primitive >
```

Example:

```
< primitive to="Hardware">
        <type>committedQoSParams </type>
        <node>
                <locator type="absolute"> inmp://192.168.1.1/</locator>
                <name>QoS</name>
                <type>Management</type>
                <capability>
                        <locator type="relative"> 00ae.4050.4350</locator>
                        <name>Cross-Layer</name>
                        <interface>Collaboration</name>
```

```
            </capability>
        </node>
        <parameter committed_value="2" measurement_unit="Mbps" >Transfer_Rate</parameter>
</ primitive >
```

Primitive *requestQoSParams*

```
<primitive to="destination_name">
        <type>requestQoSParams</type>
        <node>
            <locator type="Type"> inmp://192.168.1.1/</locator>
            <name>FC/dmFC Name</name>
            <type>Service/Management</type>
            <capability>
                <locator type="Type"> interface_identifier</locator>
                <name>Capability_name</name>
                <interface>Collaboration/Organization</name>
            </capability>
        </node>
        <parameter measurement_unit="VALUE"> Parameter_1</parameter>
        .....
        <parameter measurement_unit="VALUE" > Parameter_M</parameter>
</primitive >
```

Example:

```
< primitive to="Hardware">
        <type>requestQoSParams</type>
        <node>
            <locator type="absolute"> inmp://192.168.1.1/</locator>
            <name>QoS</name>
            <type>Management</type>
            <capability>
                <locator type="relative"> 00ae.4050.4350</locator>
                <name>Cross-Layer</name>
                <interface>Collaboration</name>
            </capability>
        </node>
        <parameter measurement_unit="Non_dimensional" >Bit_Error_Rate</parameter>
        <parameter measurement_unit="Percentage" >Bit_Error_Rate_Probability</parameter>
        <parameter measurement_unit="Mbps" >Nominal_Transfer_Rate</parameter>
        <parameter measurement_unit="dB" >Signal_to_Noise_Ratio</parameter>
</primitive >
```

Primitive *getQoSParams*

```
<primitive from="source_name">
        <type>getQoSParams</type>
         <node>
            <locator type="Type"> inmp://192.168.1.1/</locator>
            <name>FC/dmFC Name</name>
            <type>Service/Management</type>
            <capability>
```

```
                    <locator type="Type"> interface_identifier</locator>
                    <name>Capability_name</name>
                    <interface>Collaboration/Organization</name>
                </capability>
        </node>
        <parameter value="VALUE" measurement_unit="VALUE"> Parameter_1</parameter>
        .....
        <parameter value="VALUE" measurement_unit="VALUE" > Parameter_K</parameter>
</primitive >
```

Example:

```
< primitive from="Hardware">
        <type>getQoSParams</type>
        <node>
                <locator type="absolute"> inmp://192.168.1.1/</locator>
                <name>Hardware</name>
                <type>Management</type>
                <capability>
                        <locator type="Type"> 00ae.4050.4350 </locator>
                        <name>Cross-Layer</name>
                        <interface>Collaboration/</name>
                </capability>
        </node>
        <parameter value="1E-7" measurement_unit="Non_dimensional" >Bit_Error_Rate</parameter>
        <parameter value="0.2" measurement_unit="Percentage">Bit_Error_Rate_Probability</parameter>
        <parameter value="2" measurement_unit="Mbps" >Nominal_Transfer_Rate</parameter>
</primitive >
```

Message *publishQoSObject*

```
<message to="destination_name">
        <type>publishQoSObject </type>
        <node>
                <locator type="Type"> inmp://192.168.1.1/</locator>
                <name>FC/dmFC Name</name>
                <type>Service/Management</type>
                <capability>
                        <locator type="Type"> interface_identifier</locator>
                        <name>Capability_name</name>
                        <interface>Collaboration/Organization</name>
                </capability>
        </node>
        <parameter value="VALUE" measurement_unit="VALUE" > Parameter_1</parameter>
        .....
        <parameter value="VALUE" measurement_unit="VALUE" > Parameter_K</parameter>
</message>
```

Example:

```
<message to="NetInf">
        <type>publishQoSObject </type>
        <node>
```

```
                <locator type="absolute"> inmp://192.168.1.1/</locator>
                <name>QoS</name>
                <type>Service</type>
                <capability>
                        <locator type="relative">00ae.4050.4350</locator>
                        <name>Cross-Layer</name>
                        <interface>Collaboration</name>
                </capability>
        </node>
        <parameter value="1E-7" measurement_unit= "Non_dimensional"> Bit_Error_Rate </parameter>
        <parameter value="0.2" measurement_unit="Percentage">Bit_Error_Rate_Probability</parameter>
        <parameter value="2" measurement_unit="Mbps" >Nominal_Transfer_Rate</parameter>
</message>
```

Message *publishCompositeMetricObject*

```
<message to="destination_name">
        <type>publishCompositeMetricObject </type>
        <node>
                <locator type="Type"> inmp://192.168.1.1/</locator>
                <name>FC/dmFC Name</name>
                <type>Service/Management</type>
                <capability>
                        <locator type="Type"> interface_identifier</locator>
                        <name>Capability_name</name>
                        <interface>Collaboration/Organization</name>
                </capability>
        </node>
        <parameter value="VALUE" measurement_unit="VALUE">CompositeMetric</parameter>
</message>
```

Example:

```
<message to="NetInf">
        <type>publishCompositeMetricObject </type>
        <node>
                <locator type="absolute"> inmp://192.168.1.1/</locator>
                <name>QoS</name>
                <type>Service</type>
                <capability>
                        <locator type="relative">00ae.4050.4350</locator>
                        <name>CompositeMetric</name>
                        <interface>Collaboration</name>
                </capability>
        </node>
        <parameter value="45213" measurement_unit="Non-dimensional"> CompositeMetric </parameter>
</message>
```

Whenever publish to NetInf or other FC, the message format might be adapted to the specifications needed at destination.

A.2 Annex on Resource Optimization

A.2.1 Top-Down Approach

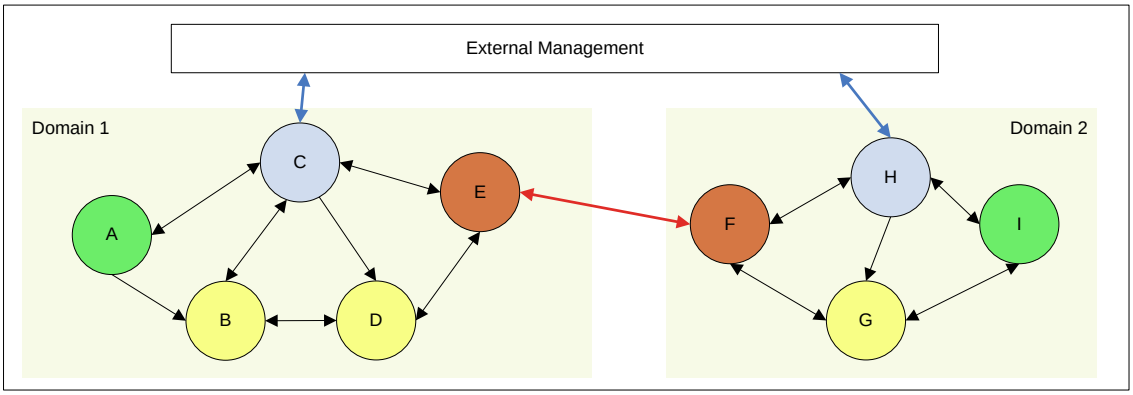


Figure A-10: Different nodes and roles inside a network

Figure A-10 illustrates a partial analysis and classification of network entities. Different types of nodes can be identified, regarding their role in the network. The behavior of a node depends on the role (or roles) it has on the network (or within a specific domain or virtual network). In this picture we can easily see how domains and boundaries affect the role of different nodes. Local (intra-domain) signalling, sharing and cooperation capabilities is required in every node (on the picture). Cross-domain interactions (inter-domain) functionalities are only required on special proxy nodes (such as E and F on the picture).

Different criterions (information source/sink, mobility, capacity, path position, distance, etc) can be used in the classification of network entities and in the identification of the possible roles performed by each entity.

A.2.2 Bottom-Up Approach

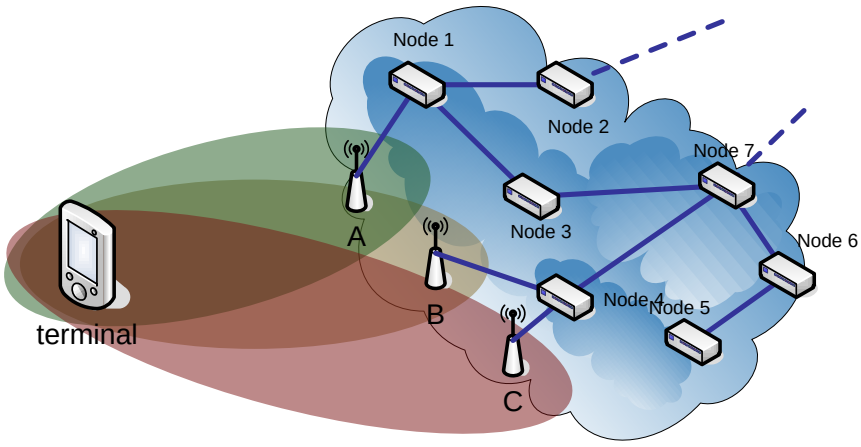


Figure A-11: Base multi-home scenario for the optimization model

Figure A-11 represents the base scenario for the multi-homing access-network selection problem. One network terminal can simultaneously reach several (three) access networks (A, B and C). This node can selected any of these networks (or all) to achieve the required services. Some of the nodes in the network will cooperate with the terminal in the selection of the best configuration. The definition of “best” depends on the requirements (user preferences, network preferences, management policies) and the status of the network.

The establishment of a local domain with the set of entities (nodes, terminal) that cooperate to solve this problem is the first phase of the process – network discovery. Identifying the relevant network parameters and how they weight on the available resources is the goal of the

model definition and will shape the necessary distributed cooperation protocol – what knowledge needs to be shared, with whom and how.

A.2.3 Optimization Process Model

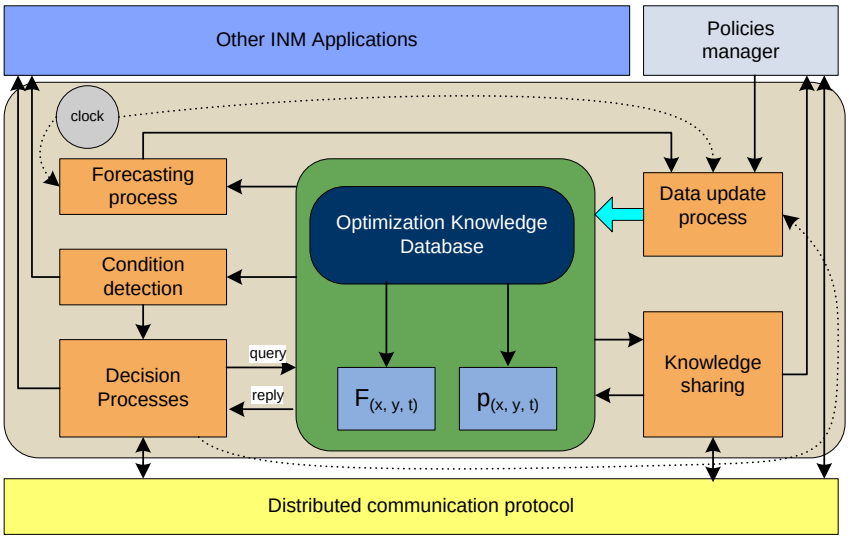


Figure A-12: Optimization process model

The Knowledge Sharing Component handles the distributed aspects for the information. It distributes local knowledge whenever necessary using the Distributed Communication Protocol. It also acquires remote information and updates the local knowledge as necessary.

Information update is triggered by internal events such as clock based periodic events or requests from other components such as the Forecasting Component. External events could also trigger the update process. The arrival of a new message (with remote knowledge information) or the definition of a new network management policy, are two examples of such events.

The Forecasting Components periodically measures the metric matrix and estimates the evolution of each relevant parameter. This estimation could be base on the shape of the variation graph of the parameter (gradient and concavity) and could arise in a probability function (represented in the picture). Periodic variation could also be detected and teach to the model by means of a transformation function (also in the picture).

The Condition Detection Component triggers an internal event whenever a parameter exceeds a predetermined range (in its simple form). This is the base mechanism for assisting other INM applications and other processes (e.g. in the control plane).

The Decision process queries the local database passing the requirement list and time as input variables. The result is a weighted matrix indicating the cost and benefit of each possible solution. The estimation of the “best” solution is obtained by a computational calculation based on a pre-determined knowledge database and a set of current requirements. This approach will allow for faster decisions, scalability of the algorithm and a small usage of computational resources, as we plan to prove. Since most of the decisions depend on the agreement of several different entities, the process is reached iteratively. After a pre-decision is made, the optimization peers (other nodes involved) are to be notified and the decision could be acknowledged or rejected. In the last case, the knowledge on the network increases and a new iteration is run.

Communication between network entities, for INM purposes, uses a general message format (INMP) and is used to exchange knowledge and decisions. A temporary domain is formed to handle a specific optimization problem. Within this domain, communication between peers is triggered and happens when:

- the domain is being created – discovery mechanisms;
- a node inquires its neighbourhood about specific (or general) information;
- a node informs its neighbours about some information (local or external);
- a node notifies its peers about some internal decision;
- a node receives a notification about some decisions from its peers;
- a network condition changes unexpectedly. The worst case scenario is if this occurs while running a decision algorithm.

Another aspect to take into account is the possible existence of partial knowledge, in which each node has incomplete information from the others, and the decisions need to be processed with this incomplete information. For this process to be feasible, a model needs to be created, and the efficiency of the decisions will be assessed according to the level of uncertainty of information.

A distributed protocol is a comprehensive set of rules that define the behaviour of each network entity. Each rule sets an action to each possible condition. An action is a “what to do”. A condition is a “when to do”.

Conditions are defined as a pair event, network state. For the same network state, different actions will take place for different event. Identically, for the same event, different actions will take place on each network state.

An event could be generated internally (clock based, condition reached, etc.) by some local process or external (network notification – alarm, received message, etc.).

A.2.4 Decentralized Traffic Matrix Estimation

We present a method to obtain traffic matrices for IP/MPLS networks. Origin-to-destination (o-d) traffic flows are calculated from measured statistical data of the Label Distribution Protocol (LDP) that is used in MPLS networks to distribute label information. The proposed method does not require the installation of any Label Switched Paths (LSPs) – routing can still be defined by IGP link metrics.

For Internet service providers there are several reasons why the knowledge of end-to-end traffic demands is crucial. The matrix of traffic demands (e.g. in kbit/s) for all combinations of two nodes (routers) within the topology under consideration is called traffic matrix (TM). In this work we aim at the calculation of average values for an end-to-end demand and for a certain time interval (e.g. five or 15 minute averages). Traffic matrices are particularly relevant and necessary in (extension-) planning and traffic engineering of IP networks. For planning network extensions traffic matrices on a point-of-presence (PoP) basis are required. They enable the simulation of various extension scenarios and the decision on topology changes: When traffic between two nodes exceeds a certain level, a direct link between those nodes might be more efficient than the extension of existing links, which would imply a higher percentage of multi-hop traffic.

Traffic engineering in IP networks includes traffic control: Routing for all demands is optimized with respect to given optimization criteria – generally Quality of Service (QoS) aspects and the overall network utilization (for example the minimization of the maximum link load) are considered [1][12]. By the introduction of Multiprotocol Label Switching (MPLS) [11] the IETF has provided a fine granular control for end-to-end traffic demands [13]: The routing for each single demand can be controlled by the definition of one or more Label Switched Paths (LSP). There are various methods to calculate and optimize LSPs (LSP-Design Problem) [6][7][8][10].

The calculation of traffic matrices of 15 minute average values for all o-d demands in a topology of 150 routers can be done in real time such that the proposed method allows for a continuous calculation of traffic matrices. Since the main prerequisite is the existence of LDP statistical data, the method is not vendor dependent but applicable to heterogeneous networks where all routers provide this information.

Traffic engineering goals can also be considered with classical routing protocols like OSPF or ISIS, that are based on shortest-path routing with respect to link metrics: The aim is to optimize the link metrics – and thus the routing indirectly – such that the given criteria are met as good as possible, see for example [3][5]. All methods of traffic engineering that aim at optimizing the routing in the network need a traffic matrix as an input. When taking different QoS classes into account, it is even necessary to have the traffic matrices for each QoS class separately. Despite the relevance of traffic matrices for network planning, there is no simple and general method so far to measure the traffic matrix in IP networks. This fact indicates insufficient activity or a weak position of network providers in the IETF standardization process and can be seen as a demand for improvement in future Internet design. An overview of methods to obtain traffic matrices for IP networks can be found in [2]. The stated methods can be grouped as follows:

Estimation of traffic matrices based on link utilizations and routing information.

Direct measurement with Cisco's Netflow approach [7].

The method we present here works for MPLS-enabled networks – more exactly for networks that use label-switching. This does not require the definition of any explicitly routed LSP (TE-LSP) – the routing can still be defined by ISIS or OSPF link metrics. In this case, paths are defined implicitly by the shortest paths. Most of the methods from [2] can also be applied to MPLS networks.

In addition, there is another alternative to obtain traffic matrices in MPLS networks: Each defined TE-LSP has usually a byte counter associated for the traffic using this LSP. Thus, building a full mesh of TE-LSPs in a MPLS network enables to measure the traffic matrix directly.

Comparison and Motivation

Existing methods for traffic matrix inference have disadvantages or might not be applicable at all in a given IP/MPLS network:

- On the one hand, the estimation from link utilization data has the advantage that it is always applicable since traffic on the links is well known and monitored by network operators. On the other hand, it is not possible to uniquely determine the traffic matrix in this way. Deterministic and statistical techniques can improve the quality of the estimation but depend on the individual network topology and additional meta data, e.g., initial estimates for the traffic matrix based on population data.
- Tracing flows in IP networks with Cisco's Netflow [3] may not be possible due to technical restrictions: not all line cards for Cisco routers support sampled Netflow and some have higher memory requirements in this case. Additionally, there is the conflict between measurement accuracy and performance degradation: the higher the sampling rate, the bigger the measurement's impact on the routing performance. Sampling rates between 1:1000 and 1:10000 can be used in practice. Also, the sheer amount of data generated by Netflow exports from all routers of a large network may induce significant costs for processing and aggregating.
- In MPLS networks the use of TE-LSP byte counters enables to measure the traffic flows between origin and destination nodes directly. The issue in this case is mainly scalability and manageability: One has to define at least one LSP per traffic demand. Many service providers use load sharing (Equal Cost Multiple Paths – ECMP) in their networks to distribute traffic more evenly. The use of one LSP per demand stops load sharing – one has to define multiple LSPs per demand explicitly to simulate load sharing with LSPs. Thus a network with nodes requires the definition and management of LSPs in the order of KN^2, where K is the mean split factor for ECMP.

```
for all Router r ∈ {1,...,n} do
  for all Path p ∈ {1,... P_r} do
    /* calculate residual paths E_{r,p} and factors F_{r,p} */
    (E_{r,p}, F_{r,p}) = getPaths (r, p, {}, 1);
    for i = 0 to size (E_{r,p}) do
      if size (E_{r,p}(i)) > 0 then
        r1 := E_{r,p}(i)(end) /* last router in path i */
        V(r, r1) := V(r, r1) + F_{r,p}(i) · B(r, p)
        if size (E_{r,p}(i)) > 1 then
          r2 := E_{r,p}(i)(0) /* second router in path */
          V(r2, r1) := V(r2, r1) − F_{r,p}(i) · B(r, p)
        end if
      end if
    end for
  end for
end for
```

Alg. 1: Calculate the Traffic Matrix

```
(E_{r,p}, F_{r,p}) = getPaths (r, p, e, f)
  E_{r,p} := {}; F_{r,p} := {};
  nextrouter := T(r, I(r, p));
  if nextrouter != 0 then
    e.push_back (nextrouter);
    if OL(r, p) != −1 then
      sum := 0;
      for all p_1 ∈ P(nextrouter, OL(r, p)) do
        sum := sum + B(nextrouter, p_1);
      end for
      for all p_1 ∈ P(nextrouter, OL(r, p)) do
        (E*_{r,p}, F*_{r,p}) = getPaths (nextrouter, p_1, e, B(nextrouter, p_1)/sum · f);
        E_{r,p}.append (E*_{r,p}); F_{r,p}.append (F*_{r,p});
      end for
    else
      nextrouter := 0
    end if
  end if
  if nextrouter == 0 then
    E_{r,p}.append(e); F_{r,p}.append(f);
  end if
return (E_{r,p}, F_{r,p});
```

Alg. 2: Calculate residual paths and factors

LDP Method

The forwarding of packets in MPLS networks is based on labels that are prepended to the regular IP header information. Packets belong to a certain *Forwarding Equivalence Class* (FEC) and as they are forwarded along a path through the network the label is exchanged at each router (and removed at the ultimate or – typically – the penultimate router of the MPLS topology. MPLS labels can be stacked, i.e. a packet can have multiple labels prepended. The label information about which label a router will use for a particular FEC can be distributed with the Label Distribution Protocol (LDP) throughout the MPLS network. Many router operating systems provide statistical data about the bytes switched in each FEC – for example the "mpls forwarding table" in Cisco's IOS [3] or the "ldp traffic statistics" in Juniper's JunOS [9]. We assume that these statistics include for each path passing through the router the incoming and outgoing labels, the FEC, the outgoing router interface and a counter for the bytes switched. In combination with knowledge of the network topology, one can determine the paths' end inside the topology. If a path begins at the router under consideration is not obvious from the existing information. The idea of the proposed method is to assume that all paths begin at the router in the first step and add the bytes switched to the corresponding element of the traffic matrix. In a second step we subtract the same amount from the traffic matrix element for the second router to the paths' destination.

Prerequisites

Given a network topology with n routers, where each router has I_r, $r = 1, \ldots, n$ interfaces. The network topology is represented by the function T that assigns to each pair of router and interface either its destination router or 0, if this interface leads to a router that does not belong to the topology under consideration

$$j \in \{1, \ldots, n\} \text{ and } i \in \{1, \ldots, I_r\}: \; T(r, i) \in \{0, 1, \ldots, n\}.$$

We denote by P_j the number of paths that pass through router j that has the following information for each path p:

- $I(r, p)$: outgoing interface of path p at router r,
- $B(r, p)$: traffic that was switched on path p at router r during a defined time interval,
- $IL(r, p)$: incoming label of path p at router r,

- $OL(r, p)$: outgoing label of path p at router r.

The following aspects of label switching have to be considered when calculating traffic matrices in the proposed way:

- In networks with load sharing, one router can split paths so that one incoming label is mapped to multiple outgoing labels (and multiple outgoing interfaces). By $P(r, il)$ we denote the set of paths that pass through router r and have the same incoming label il. One cannot rule out the possibility that the new paths end at different routers of our given topology. Although this might seem improbable for complete topologies, the more general approach can be useful when calculating traffic matrices for subsets of real MPLS networks. Thus, our method does not make any assumptions that paths that are split on the way end at same node of the network.
- The reverse case might also occur: packets arrive with the same label from different routers and the paths are merged with one forwarding rule to single one.
- If the two cases mentioned above occur jointly, i.e. packets arrive from different routers with same labels and are split to several outgoing paths, we assume that all incoming traffic is split with the same ratio.
- As stated before, labels can be stacked and it can happen that a label is not exchanged but removed at a router. This is indicated by special keywords in the LDP statistics (e.g. “Pop Tag” instead of an outgoing label in Cisco’s MPLS forwarding tables). We assume that the paths ends at the next router, if belonging to our topology and set in this case $OL(r, p) = -1$.

Algorithm

With the information described in Section 3.1, a “forward-linking” of flows is given and we can calculate for each path p that passes through a router r its endpoints in our topology. We denote by $E_{r,p}$ the list of *residual paths*. Each element of $E_{r,p}$ represents a list of routers that connect r with its endpoints for path p and by $F_{r,p}$, the *list of* corresponding splitting factors. Each element of $F_{r,p}$ is the part of the traffic of path p that uses the corresponding residual path. We have $\sum_k F_{r,p}(k) = 1$.

The residual paths and splitting factors can be calculated recursively with algorithm 2. In case that r is the first router of path p we can add the traffic to the corresponding elements of the traffic matrix. Otherwise those elements are increased by mistake. Since all routers in the topology are treated, this error can be corrected: For the paths second routers we subtract the traffic of the corresponding elements of the traffic matrix, if the second router is not the last one. In this way, all traffic added by mistake is corrected, because each router that is not the first in a path, is the second for some other router of the topology. More precisely, this can be implemented as shown in algorithm 1.

Figure 1 illustrates the stepwise execution of the algorithm for a topology of four routers and two flows each of which transporting 10 units of traffic. When processing router 1, the traffic matrix element TM(1, 4) is increased, while TM(2, 4) and TM(3, 4) are decreased. In later steps [2] and [3] in figure 1, the elements TM(2, 4) and TM(3, 4) are then set back to zero.

Results

The proposed method has been successfully deployed in Deutsche Telekom’s global MPLS/IP Backbone network [13]. Even for topologies of more than 150 nodes, the calculation of traffic matrices for 15-minute average values could be carried out in real time, i.e. in less than one measurement interval on a commodity PC such that a continuous calculation was possible. The efficiency of the calculation also reduces the amount of data that has to be stored: The collected raw data doesn’t have to be stored over a longer period, but only the calculated traffic matrices.

Conclusions

The presented method to calculate traffic matrices in MPLS networks only requires the existence of a label based switching and a traffic accounting on this scale. The method does not depend on the way routing is controlled (e.g. explicit LSPs or shortest path routing based on IGP metrics). Apart from the abstract algorithm, we discussed the application in real life MPLS networks. At the moment there are some restrictions that result from incomplete availability of the statistical data and vary with the topology under consideration and the router vendor. On the other hand, for a given network the relevance of these restrictions can be easily analysed a priori. If they do not apply the method provides accurate traffic matrices for MPLS networks and is easy to implement.

Local tag	Outgoing tag or VC	Prefix or Tunnel Id	Bytes tag switched	Outgoing interface
12313	Pop tag	1.2.3.4/30	11344	P01/0
12315	12337	1.2.3.6/32	243350	P01/0
	12326	1.2.3.6/32	237605	P05/0
12316	12338	1.2.2.2/32	374291	P01/0
	12327	1.2.2.2/32	286587	P05/0

Table A-1: Output of "show mpls forwarding-table" in Cisco's IOS [3]

FEC	Type	Packets	Bytes	Shared
1.2.3.4/32	Transit	48239	234242	No
	Ingress	32122	12312	No
1.2.3.5/32	Transit	123123	1057084	Yes
	Ingress	543454	243350	No
1.2.3.6/32	Transit	34323	237605	No
	Ingress	343	374291	No

Table A-2: Output of "show ldp traffic-statistics" in Juniper's JunOS [9]

[1] D. Awduche, A.Chiu, A. Elwalid, I. Widjaja and X. Xiao: Overview and Principles of Internet Traffic Engineering, IETF RFC 3272 (2002)

[2] N. Benameur, J.W. Roberts: Traffic Matrix Inference in IP Networks. Proc. Networks2002, VDE-Verlag, 2002, pp. 151-153

[3] Cisco Systems, Netflow Aggregation Cisco IOS release 12.0(5)T <www.cisco.com>

[4] B. Fortz, M.Thorup: Internet Traffic Engineering by Optimizing OSPF Weights. Proc. IEEE Infocom, March 2000.

[5] B.Fortz, M. Thorup: Robust optimisation of OSPF/IS-IS weights, Proc. INOC (2003) 225-230

[6] M. Franzke, A. Pönitz: Global shortest path solutions for the traffic engineering problem, Proc. Networks, VDE-Verlag, (2002) 275-278

[7] G. Hasslinger, S. Schnitter and M. Franzke, The Efficiency of Traffic Engineering with Regard to Link Failure Resilience, Telecommunication Systems 29, 109-130 (2005)

[8] G. Hasslinger, S. Schnitter and M. Franzke, TE-Scout: An Offline Traffic Engineering Tool for Optimised Path Design in IP Service Provider Networks, Proc. MMB Conf., Dresden, Germany (2004) 35-44

[9] Juniper Networks, Junos OS documentation <www.juniper.net>

[10] M. Pioro and D. Medhi, Routing, Flow and Capacity Design in Communication and Computer Networks, Morgan Kaufmann Publishers (2004)

[11] E. Rosen, A. Viswanathan, R. Callon, Multiprotocol Label Switching Architecture, IETF RFC 3031 (2001)

[12] S. Schnitter, F. Hartleb and M. Horneffer, Quality-of-service class specific traffic matrices in IP/MPLS networks. Proc. Internet Measurement Conference (2007) 253-258

[13] S. Schnitter, A. Barth and O. Schnitter, Benefits from 2-Layer Traffic Engineering for IP/MPLS Networks, NETWORKS Conference <www.networks2008.org>, Budapest, Hungary (Sept. 2008)

A.2.5 Inaccuracy Problems for Situation Awareness Using Standard OLSR Routing

Another case study is focused on using the optimized link state routing (OLSR, RFC 3626, www.ietf.org/rfc/rfc3626.txt) as the standard routing protocol of the MANET working group at the IETF for situation awareness. We consider wireless broadcasting nodes placed at different locations spread over an area even without mobility. We assume that two nodes can directly exchange messages when their distance does not exceed a common transmission range. In this way, the transmission links between nodes and the network topology is determined by the node locations.

The nodes route messages via OLSR over multiple hops, including Hello and topology control (TC) messages at default intervals of 2 and 5 seconds, respectively. We consider the queue size of messages that arrived at a node but are not yet forwarded towards the destination as a QoS measure. OLSR is extended to distribute the queue size such that each node is aware of the queue size at all other nodes subject to a delay until routing updates are received between nodes. The queue size information is not used for load balancing. To store the QoS-related state associated with a node, a new field is added to the neighbourhood information base and to the topology information base maintained by the protocol. To populate these fields, the message format of Hello and TC messages were extended as well. Table 111 summarizes the parameters of the modeling and simulation parameters.

In this scenario, we study the absolute difference between the current queue size at a node and the aged information about it, which is available at the other nodes through routing messages at the same time, as a measure for inaccuracy of the routing information state.

Figure A-Figure A-13 combines the evaluation of deviations in queue size information with the age of the information at the nodes. The *k*-th column gives the mean deviation for nodes whose information is aged between *k*-1 and *k* seconds or is larger than 21s in the last column.

Simulator Parameters	
Network Type	IEEE 802.11
Propagation model	Two-Ray-Ground
Mobility model	Static
Transmission range	250 m
Network topology	50 nodes randomly located in 1 km^2
Traffic model	20 random source-destination pairs, constant messaging intervals: 0.2s, 0.14s, 0.09s, 0.04s, 0.02s for different traffic load levels
Packet size	128 Byte
Queue	for max. 50 packets; Tail-Drop
Simulation time	200s; 50s start phase not evaluated

Table A-3: Simulation Parameters

As can be expected, the deviation is essentially increasing with the traffic load, which varies in

the five curves by a factor of 10 from low and medium load up to congestion. Inaccuracy is also increasing with knowledge age, although not monotonically and partly even decreasing for the low load curves. This effect is due to the fact that the paths of messages utilize nodes in the middle of the square area more often than nodes in the outer regions. Thus the nodes in the middle experience higher load and queue variability. But the knowledge age about nodes in the outer regions is larger, while they have a low and less variable queue size. This trade-off causes a tendency to decrease the inaccuracy in spite of larger knowledge age, which becomes apparent especially for the low traffic curves.

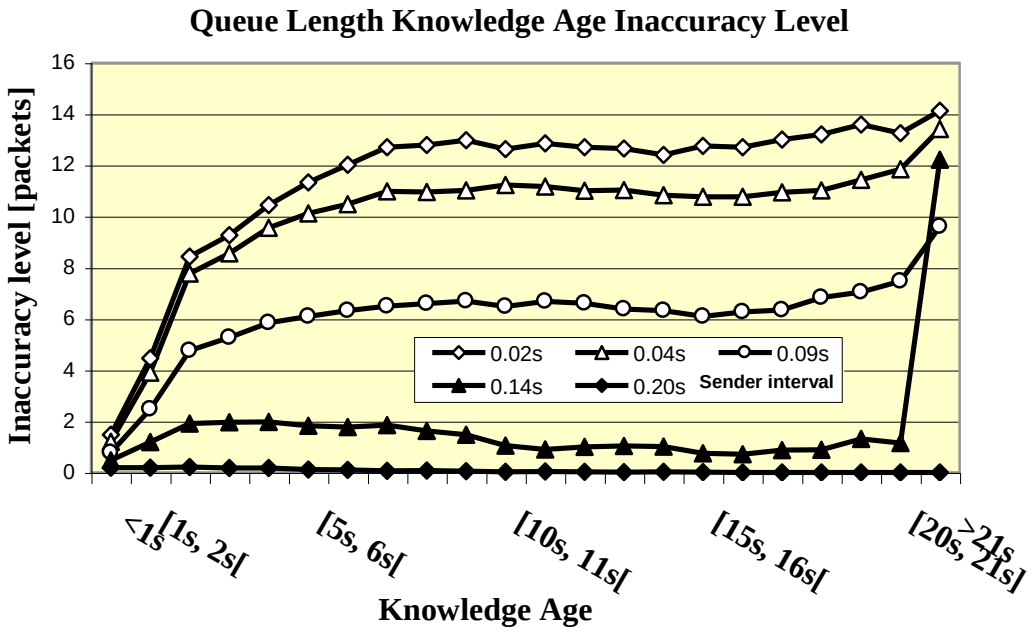


Figure A-13: Inaccuracy in routing information for situation awareness

We also analyzed the impact of sending more frequent Hello and TC messages but did not obtain a positive impact on the overall inaccuracy level. In order to reduce knowledge age, we enhanced the routing by a probing scheme. The current state of a node is looked up by a probing message which is triggered by a threshold for the age of information. Probe messages are fully exploited by updating the new status information for all nodes on their paths as well as for nodes which receive the information due to broadcasting in the surrounding of the path. The probing scheme was able to reduce the mean knowledge age to about half, but in spite of the improvement in knowledge age, probing was unable to reduce the inaccuracy in the queue size estimates. This is also apparent from the results in Figure A-13, since the inaccuracy level stays almost constant for knowledge ages of 5 and more. Only for ages below 3 seconds essential improvements of the accuracy are observed.

This case study shows that inaccuracy of routing information can be high in many wireless and mobile communication scenarios. Inaccuracy referring to mobility or high churn instead of queue sizes in the considered fixed topology will also affect the transport of routing messages which may cause even more inaccuracy due to route flapping and loops.

When high dynamics impedes convergence of link state and distance vector routing protocols, then flooding and random walks can be used as well as methods combining flooding, random walks with partial routing knowledge [HaKu].

[HaKu] G. Hasslinger and T. Kunz, Efficiency of search methods in dynamic wireless networks, Proc. Wireless and Mobility, Barcelona, Spain, Springer LNCS 5122 (2008) 142-156

B Cooperation Strategies for Anomaly Detection (in Self-Protecting Networks)

As strategies are being developed that benefit of the collaboration of network entities to achieve superior performance in self-protection and self-management. Challenges are tackled by the components which are best suited to the respective task. Self-management tasks disburden the administrators of the network by allowing controlling heterogeneous components using high-level policies.

Self-management implies that the network can anticipate, detect, identify and protect against threats that disrupt normal operation. Self-protecting components detect abnormal hostile behaviour and take corrective actions to become less vulnerable; they implement a control loop for anomaly detection and triggering of countermeasures.

Collaborative self-protection relies upon distributed detection of anomalies to gain significantly in oversight and specialization. The cooperation among the nodes adds a dimension that can be exploited to enhance the detection capabilities of the network.

Cooperation for self-protection extends towards aspects of

- Task distribution
- Information sharing
- Cooperative joint decision making

To address these points, we investigated concepts for collaborative self-protection of networks. We have then exemplified the challenges involved in addressing a typical network scenario; the protection against distributed denial of service attacks.

Bandwidth is a scarce resource which must be diligently and fairly assigned to the legitimate users of a system. While usage calculations help to invest in bandwidth resources with relative success, phenomena like flash crowds, abusive and malicious usage may diminish the availability to our services, or even result in a complete Denial of Service (DoS). An overload of system or network resources commonly renders services completely unavailable. DoS attacks account for some of the largest and highly visible damages in the Internet of today. Administrative measures to prevent such behaviour from the user side are difficult, if not impossible to take. Mitigation attempts may always exclude regular customers from the service. A sacrifice of functionality and comfort is the price to be paid for applying additional security policies. To maintain a reasonable security/functionality equilibrium is not an easy task.

The collaborative approach we present to this problem is to leverage the specialization of the protected services on the individual protocols they process. Services and components in the network may report beneficial and malicious usage to reactive components in the network. These components react by shaping or filtering the traffic according to the valuation of its services. In other words, the approach is to mitigate bandwidth overload independently of its causes while maintaining a high availability of the service. The system should react in case of resource consuming behaviours and conditions without distinguishing between attackers or legal requests. The objective is to manage and maintain the service available for the target audience. In this sense, if our service is available as we want it to be, we can manage it and be content.

The approach consists of a cooperation of three types of component. The protected services provide valuations for their users. These valuations are mapped to an uniform range per service, which we define as $[-1.0; 1.0] \in R$. Semantically, this corresponds to the usefulness of the user for the service or business objective. Hence, every service can only provide a subjective classification of its users. These numerical ranges are easily aggregated, weighted, and forwarded to other components without losing their inherent meaning. A website may serve as example here: A user visiting the site with low frequency might get a slightly positive rating. A site scraper with high frequency a slightly negative one. A successful login with a

user account receives a more positive rating. Successfully purchasing a product receives an even higher rank.

These observations will be reported to either a central Management Controller (MC) component, or a multiplicity of controllers whose major task is to collect and process this information. At the same time, the Controller receives traffic statistics from a measurement component on a central router. The observed metrics are: The total volume of traffic passing the router; and the volume of traffic per network flow reaching the system. The first metric allows the Controller to detect if the processing capacity of the system is about to be exhausted. A threshold could be set at a sufficiently large fraction, e.g. to 95% of the traffic that can be safely processed.

The second metric allows to automatically generating rules in order to protect the services against overloading. In the case of overload, the Controller will generate filters on these users which have the lowest aggregated rating across all services, while estimating the saved bandwidth using the bandwidth per user metric. This process continues until the estimated total bandwidth after filtering is below the indicated threshold. In order to restore normal operation after an overload, the filters may be slowly eroded when the total traffic reaches a sufficiently low level.

C Extended Comparison of 4WARD INM with Related Projects in EU and US

ANA – Autonomic Network Architecture

http://www.ana-project.org/

The goal of this project is to design and develop a novel network architecture beyond legacy Internet technology that can demonstrate the feasibility and properties of autonomic networking. The ANA architecture will facilitate self-* features such as self-configuration, self-optimisation, self-monitoring, self-management, self-repair, and self-protection.

This work aims at including monitoring as an integral part of the network architecture. This brings a set of new requirements: monitoring needs to be dynamic, adaptive and programmable. In contrast to traditional approaches, monitoring must not assume a priori knowledge about the network itself, but instead monitoring functions may be placed and self-configured dynamically in the network. For this to happen it is required that modules explore the available monitoring support in their environment at runtime.

Besides performing conceptual work, ANA puts emphasis on prototypical realisation. The architecture relies on the paradigm of functional composition of modules at runtime. A number of functional blocks are currently under development, e.g., packet capturing, sampling and system monitoring.

The problems addressed by ANA are close to those addressed by 4WARD. Both projects aim at increasing the level of network automation. Nevertheless ANA should be regarded as a generic architecture for autonomic devices, while InNet Management will leverage on a tight coupling of management functions with the services deployed on a device, like virtualisation of resources or generic paths.

For 4WARD, it is of particular interest the work in ANA on design principles, mechanisms and proof-of-concepts for autonomic network management. This includes functions for monitoring, self-optimisation and resilience.

E3

https://ict-e3.eu/

E3 is aiming at a gradual, non-disruptive evolution of current wireless systems into an integrated, scalable and efficiently managed B3G cognitive radio system framework. E3 will optimise the use of the radio resources and spectrum, following cognitive radio and cognitive

network paradigms characterized by reconfigurable, self-adaptive, autonomic and collaborative features.

Areas of work comprise

- identification of market opportunities, business cases and regulation issues;
- definition of the Cognitive Radio Architecture including the definition of generic functional blocks for cognitive radio management;
- collaborative cognitive radio resource management, spectrum management & self-organization with focus on network-based decisions;
- autonomous functionalities & algorithms focusing on terminal-based decisions (e.g. access of terminals to “wireless highways”);
- supporting functions for heterogeneous standards defining enablers for cognitive operation, (e.g. Cognitive Pilot Channel (CPC) to support an efficient discovery of the available radio accesses and to help in reconfiguration management in heterogeneous wireless environment between network and user terminals)
- prototyping environment for validation of E3 concepts.

Management functions in E3 will be distributed over different network elements.

Both, E3 and 4WARD aim at similar goals, namely designing autonomic management systems, i.e., management systems with self-* functionality. The difference between the projects is that E3 focus on wireless scenarios, while 4WARD has a wider scope in terms of considered scenarios.

AutoI - The Autonomic Internet

http://www.ist-autoi.eu/

This project proposes to transition from a service agnostic Internet to a service-aware network where resources are managed by applying autonomic principles. The key premise behind AutoI is that networks today have no awareness of services running over them. While there are some nodes that support QoS solutions, this is far from uniform with the vast majority of nodes.

The approach to network management in AutoI is catered for in one of the 5 planes of the AutoI architecture, which are Virtualisation, Orchestration, Management, Service Enabler and Knowledge planes. The Management Plane promotes in-network management where the management system is distributed across the routing nodes in the Internet. Specifically, a policy based autonomic control loop exists for each node that monitors, decides and act accordingly in order to support self-FCAPS functionality. Inter domain aspects in AutoI are catered for using the Orchestration Plane that can handle federation, negotiation and distribution between the autonomic managed domains (named AMS in AutoI). Therefore each AMS performs management with local objectives, but can itself be managed to abide by more global objectives via orchestration.

Similarly to 4WARD, this project aims at developing a distributed self-managing management plane. However, unlike 4WARD, this project focuses on virtual resources.

Simple Economic Management Approaches of Overlay Traffic in Heterogeneous Internet Topologies

http://www.smoothit.org/

SmoothIT addresses innovatively the detailed economic and technical mechanisms for a flexible, secure, and scalable traffic management of overlay networks in tomorrow's ISPs and telecommunication operators networking infrastructure.

For today's Telecommunication Service Providers (telco) and Internet Service Providers (ISP) the issue arising is: how to control and manage network traffic stemming from overlay-based applications. As the structure of overlays determines the traffic flows in ISP networks, it is highly efficient for an ISP to influence overlay configuration based on information on their structure. Overlays have to be managed to maximize the benefit for multiple operators/ISPs involved, and to increase the capability to withstand faults and balance the network load.

Therefore, SmoothIT pursues the following major objectives:

- To structure overlays in a way that is efficient or optimal, both for user communities and for ISPs. This is to be attained by means of incentive mechanisms.
- To study and define key requirements for a commercial application of Economic Traffic Management (ETM) schemes for ISPs and telcos.
- In order to advance traffic management beyond traditional limits, specialized economic theory will be applied for building in a fully decentralized way network efficient Internet-based overlay services in multi-domain scenarios, solving the information asymmetry problem.
- To design, prototype, and validate the necessary networking infrastructure and their components for an efficient implementation of such economic traffic management mechanisms in an IP testbed and trial network.
- To develop an optimized incentive-driven signaling approach for defining (theory) and delivering (technology) economic signals across domain boundaries in support of co-operating and competing providers in an interconnected heterogeneous network environment.
- To stress operator-orientation by verifying key results of the work through ISP and telco requirements as well as its supporting technology.

4WARD is broader in scope and is obviously related to SmoothIT in several ways. Overlay networks as a main topic of SmoothIT represent concrete scenarios within the more abstract and generalized virtualization framework to be developed by 4WARD. Traffic management is addressed in SmoothIT by similar distributed and self-organizing concepts as the broader InNetworkManagement (INM) approach of 4WARD. Traffic management schemes are in the focus of both EU projects, where the focus of SmoothIT on economic and game theoretical view of cooperative and competing multi-provider behaviour is a supplement to the study of advanced technical approaches for network management architecture and algorithms in WP4 of 4WARD. Obviously there is much potential for both projects to profit from each other.

Architectural Support for Network Trouble-Shooting

http://www.nets-find.net/Funded/ArchtSupportNet.php

This project focuses on troubleshooting in networked systems, including failure detection, identification, root-cause analysis and attributing problems to responsible parties. The project team argues that while troubleshooting is a crucial and fundamental problem, the current architecture provides very little support to perform this in an efficient manner. They therefore propose to annotate network traffic with meta-information such as the role, path, or manifestation of a packet. This would then facilitate causality tracking in the network in terms of how instances of network activity relates to earlier activities. The team also intends to study more elaborate forms of logging of network activity, in which the logging entities have an active role in requesting information from generating entities, and how to construct aggregated views of several logs in certain repositories.

Although this project and 4WARD have the common goal of achieving efficient troubleshooting, the approaches are different. 4WARD does not consider annotating network traffic for troubleshooting, but instead focuses on algorithms that do not necessarily need such data. Also, 4WARD mainly studies real-time in-network management using distributed

approaches, while this project focus more on centralized solutions for studying historical data. Still, as the possibility of annotating network traffic could have implications on the algorithms developed within 4WARD, it is of some interest to study the results of this project.

A Framework for Manageability in Future Routing Systems

http://www.nets-find.net/Funded/Framework.php

The goal of this project is to develop a framework for specifying, understanding, and evaluating what features should/could be "designed-in" into routing systems in support of manageability and to evaluate design choices and trade-offs thereof in terms of performance and manageability. Routing manageability is the ability of routing to monitor, control, and trouble-shoot its operation. Manageability is based on:

- Sensing capability - monitoring and detecting of changes in the network state and association of decision rules and actions taken by a routing element with the appropriate events in the network state.
- Logging and Reporting capability – functions to locally collect and record visibility information. The reporting implementation is seen as in-band or out-of-band.
- Change or Event Notification capability – publish/subscribe like functionality related to the routing management reports
- Querying capability – is the ability for another entity to query a routing element for certain information.
- Real-time Actuation capability – real-time actions required to take when some pre-specified conditions are met.

Finally, using the described above functionality a manageable distributed computation routing protocol will be developed. It will encompass distributed loop-free path computations.

The vision pursued by this project is in line with that of 4WARD. Therefore, we intend to closely study the approach used by this project and their results.

Design for Manageability in the Next Generation Internet

http://www.nets-find.net/Funded/Manageability.php

The objectives of this project are to explore technology design alternatives for network management in Next Generation Internet (NGI), and produce an educational and outreach plan, enhancing networking curriculum. The project focuses on low-level building blocks that can be composed in different configurations to enable an Internet management plane. Examples of such blocks include protocols for data sharing and event detection mechanisms. The project emphasizes prototype development. It uses large-scale test-bed, and produces methodology for experimenting the design alternative, in addition to design documentation and some initial prototypes for the building blocks, as well as an educational plan and course material.

While the project addresses NGI, it is not a complete clean slate approach, as 4WARD is. It addresses the evolutionary steps required to meet future demands. It assumes existing infrastructure and technologies, such as optical circuit-switching core networks, packet-switching access networks, and OSI layered protocols.

Similarly to 4WARD, this project aims at designing automated and intrinsic management solutions. The goals of both projects differ in that this project puts more emphasis on test-bed experimentation methodology. A second difference is that this project also considers educational and outreach plans.

Model-based diagnosis in the Knowledge Plane

http://www.nets-find.net/Funded/ModelBased.php

The goal of this project is to develop a Knowledge Plane (KP) that serves as a component of a Future Internet, which is self-managing and self-diagnosing. The KP locates, collects, and manages existing knowledge in a distributed way and specifies a framework that supports queries to get that knowledge. This effort includes the development and documentation of a high-level architectural framework for the KP and the identification of collaborators with expertise in specific problem areas that are the target customers of the KP architecture.

The goals of this project and Task 4.3 in 4WARD overlap partially, for they aim at collecting and offering information to decision-making processes. Hence, we intend to closely study the results achieved within this project.

A Network-Wide Hashing Infrastructure for Measurement and Monitoring

http://www.nets-find.net/Funded/NetworkWide.php

The focus of this project is network monitoring. The goal of the project team is to engineer monitoring systems that help improving our understanding of network behaviour. They argue that this is not possible with current monitoring systems. Therefore, they advocate for a clean slate approach to network monitoring, and their vision consists in all network elements having a generic functionality for creating sketches of local metrics. This functionality uses hash functions and controls the trade-off between local memory space and monitoring accuracy. The project also considers creating network-wide views by aggregating local sketches.

The goals of 4WARD and this project overlap. Both projects, for instance, develop algorithms for network monitoring with controllable performance. The specific techniques considered by each project are different though (hash functions vs. aggregation functions). While 4WARD has a strong focus on real-time monitoring, this FIND project considers historic data as well. As some WP4 work is complementary to this FIND project, we intend to closely study the results achieved within this project.

Towards Complexity-Oblivious Network Management

http://www.nets-find.net/Funded/TowardsComplexity.php

This project proposes the creation of a Complexity-oblivious Network Management (CONMan) in order to simplify network management. Two sources of complexity are cited. First, the current management plane depends on the data plane. An example cited is as follows "SNMP operates on top of the data plane and hence, management protocols rely on the correct operation of the very thing they are supposed to manage." The second source is the growing complexity of networks. The example cited in this case shows that relatively simple network devices can have many thousands of management objects.

It is proposed to reduce the complexity of network management by using new and novel techniques. Firstly, all data plane protocols must expose a generic complexity oblivious management interface. Management operations then get carried out in a distributed fashion by software network managers located on network devices. These are managed by policies which originate as a high-level domain specific language. This is broken down from goals to specification by these software network managers. The architecture, design and implementation of the software is a framework for new network management objects. This bears many similarities to 4WARD. As implied by the use of 'new and novel techniques', there is a certain parallel with the clean slate approach adopted by 4WARD of using 'novel techniques'. The project also proposes that data plane protocols should all expose a generic management interface. This is extended in the work of 4WARD by not only specifying data plane protocols. The notion of management operations being carried

out by network managers located on distributed network devices is also developed in 4WARD with self managing entities and functional components where the mapping to nodes is not explicitly stated. While not completed yet, 4WARD has already considered the idea of breaking high-level 'governance' into low-level SLAs to achieve a goal to specification mapping using the SE's and FC's. Finally, the style of architecture, design and implementation through a technical framework is a common trait between both projects. As both, this project and 4WARD have similar goals and propose similar approaches, we intend to closely study the results achieved within this project.