Architectural Principles and Elements of In-Network Management

Dominique Dudkowski*, Marcus Brunner*, Giorgio Nunzi*, Chiara Mingardi*,
Chris Foley[†], Miguel Ponce de Leon[†], Catalin Meirosu[‡], and Susanne Engberg[‡]
*NEC Laboratories Europe, Network Research Division, Heidelberg, Germany
Email: {dominique.dudkowski|marcus.brunner|giorgio.nunzi|chiara.mingardi}@nw.neclab.eu
[†]Telecommunications Software & Systems Group, Waterford Institute of Technology, Waterford, Ireland
Email: {ccfoley|miguelpdl}@tssg.org
[‡]Ericsson Research, Ericsson AB, Stockholm, Sweden
Email: {catalin.meirosu|susanne.engberg}@ericsson.com

***Abstract*— Recent endeavors in addressing the challenges of the current and future Internet pursue a *clean slate design* methodology. Simultaneously, it is argued that the Internet is unlikely to be changed in one fell swoop and that its next generation requires an *evolutionary design* approach. Recognizing both positions, we claim that cleanness and evolution are not mutually exclusive, but rather complementary and indispensable properties for sustainable management in the future Internet.**

In this paper we propose the *in-network management* (INM) paradigm, which adopts a clean slate design approach to the management of future communication networks that is brought about by evolutionary design principles. The proposed paradigm builds on embedded management capabilities to address the intrinsic nature, and hence, close relationship between the network and its management. At the same time, INM assists in the gradual adoption of embedded self-managing processes to progressively achieve adequate and practical degrees of INM. We demonstrate how INM can be exploited in current and future network management by its application to P2P networks.

***Index Terms*— clean slate design, evolutionary design, in-network management, self-management, future Internet**

I. Introduction

ALTHOUGH management is considered an inseparable part of communication networks, its intrinsic nature is not reflected in current networks. Instead, a clear separation exists where network functionality is designed and deployed before management is superimposed as an add-on feature. This clear mismatch has lead to a number of significant and growing problems: the incremental adding of features and the patch-on-a-patch approach apparent in many of today's systems have resulted in large complexity, nonscalability, and the need for extensive manual involvement. Trends in how the properties of current networks will develop indicate that management based on conventional paradigms will eventually break down, because the incremental adding of management features will become impractical.

Whereas the need for structural change is broadly acknowledged, there is large controversy about the best way to bring about the necessary changes. Two Internet design methodologies dominate in recent EU- and US-funded initiatives: *clean slate* (e.g. Nth stratum [1] within 4WARD [2] and 4D [3]) versus *evolutionary* architectural design [4]. While clean slate considers what future networks would look like if one started from scratch, proponents of evolutionary design argue for nondisruptive transitions over time [4]. We believe that *both* views in combination constitute an essential and immutable principle for the development of future communication networks, and specifically, the Internet.

We claim that this view holds for future network management in particular. Within this scope, we are unaware of any previous methodology that suggests how the gradual implementation of a clean slate management design can be made practical. To this end, we propose the *in-network management* (INM) paradigm, which combines the clean slate and evolutionary design principles to achieve sustainable management in future communication networks.

Rather than focusing on a single design approach, in-network management provides the necessary concepts and procedures to induce gradual changes in the way management is done today. While defining a pure case of INM, where the intrinsic nature of management is reflected in the network architecture, INM allows adapting sensible and practical degrees of embedding management functionality, in different locations of the network. Thereby, the adoption of a clean slate design is brought forward by an evolutionary process with a clear, flexible, yet tangible goal.

In-network management provides concrete architectural concepts that facilitate the embedding of management functionalities inside the network and network elements. INM does not shift complexity by proposing a generic solution, but provides fundamental management capabilities that may be combined to capture dedicated and clear base management functionality with sensible complexity, including FCAPS functions. While incorporating legacy management systems in the evolutionary dimension is supported, INM shows most of its benefits if applied to a fairly novel way of designing future networks, specifically, in those cases where the network management functionality is inherently designed into the future Internet's functions and protocols.

The remainder of this paper is structured as follows. After reviewing related work in Sec. II, we introduce the paradigm and underlying principles of in-network management in Sec. III. The architectural elements of INM are described in detail in Sec. IV. To show how in-network management can be exploited and adopted we illustrate its application to Peer-to-Peer networks in Sec. V. We conclude with a brief summary and outlook to future work in Sec. VI.

II. Related Work

A growing number of future network research initiatives are dealing with questions on how network management is to be accomplished. The authors of [3] move from the consideration that today's management functions need to be mapped to the logic elements of the nodes. As a consequence, they propose a clean slate approach and introduce four separate planes: decision, dissemination, discovery, and data. While we agree with the initial analysis, we observe that the approach closely resembles existing telecom architectures with dedicated channels and machines for specific management functions. Furthermore, such separation would have the effect of shifting the complexity to the proposed planes. The authors also seem to ignore the cost associated with the implementation of separate planes. The architecture in [5] is built with similar principles. Management operations are defined over a general interface, which is instantiated within each protocol entity. The architecture sounds valid, but authors do not mention the complexity in the instantiation of functions from their general interface. The authors of the clean-slate architecture in [6] consider the technical challenges of function composition. These considerations are somewhat general for any composition framework and can certainly be considered in INM, at least for those aspects related to composition.

The problem of mapping high-level objectives into service-specific settings is presented as a key issue in [7]. The concept of a pervasive knowledge plane is introduced, where artificial intelligence and cognitive systems are enabling techniques, but authors do not go further in defining functional elements yet. Within the PlanetLab project, an information plane is presented in [8], where network elements are reconfigured by declarative programs. Self-management is instead the main objective of the knowledge plane proposed in [9], which is achieved through collaborative and autonomous multi-agent systems that are embedded within network elements. The above works testify that aspects related to knowledge and information play an important role in management operations and they can be considered as building elements in the definition of a new architecture. Nevertheless, mechanisms for knowledge distribution can follow different degrees of embedding, and therefore also mapped into our proposal.

The management architecture in [10] proposes a model representing a given network aspect, stating the conditions for executing a function according to policies. The architecture targets mainly wireless sensor networks, with a restricted set of issues. The FOCALE architecture in [11] emphasizes the use of information and ontological modeling to gather knowledge about network capabilities. The system is highly autonomous, but it is very complex and therefore difficult to understand in case of unforeseen failures in the management system itself. INM's goal is rather the design of an autonomous system that is kept simple and flexible, providing a balanced level of autonomy and abstract interfaces to allow interactivity with the system. The ASA architecture in [12] aims at enabling autonomic management of resources to guarantee Service Level Agreements (SLA). ASA's main advantage is that it encompasses different abstraction layers and heterogeneous resources. However, it is characterized by large complexity, both in the hierarchical structure of the management entities and in the internal structure of such entities.

The Ambient Networks project [13] supports composition of networks across business and technology boundaries. While it provides valid mechanisms for composition, the mapping between different control spaces is still a manual step of the process. Madeira [14] proposes a distributed management system with self-forming logical overlay topologies. It is mainly targeted at wireless networks and still adopts a hierarchical structure with middle managers. ANA [15] is building an architecture that can demonstrate the feasibility and properties of autonomic networking. The problem field is quite close to the topics addressed in INM, but ANA should be regarded as a generic architecture for autonomic devices, while INM will leverage on a tight coupling of management functions with the services deployed on a device, like virtualization of resources or generic paths. Furthermore, ANA has a strong emphasis on prototypical realization.

In contrast to the discussed approaches, we propose a clean slate approach to future network management that can be gradually achieved by evolutionary processes. We further present the necessary architectural principles, elements, and methodology in order to do so, and demonstrate how this can be achieved by the concrete example of a P2P system.

III. Principles of In-Network Management

INM addresses the challenges of current and future network management by combining a clean slate design paradigm with an evolutionary design methodology. In order to achieve this objective, INM firstly stipulates five fundamental principles, which capture, in our view, the essence of future management of communication networks. The first principle addresses the very nature of network management per se:

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

The following three principles are consequences from the intrinsic principle and define the extremal *clean slate* architectural design of in-network management. We note that these principles are extremal cases that will be relaxed in our subsequent practical considerations:

2. Inherent principle: Management is an inherent part of network elements, protocols, and services. This principle captures the most extreme version of the architectural design where management functionality is coalesced with the rest of the network functionality. As such, management becomes an inseparable and indistinguishable part of the network, thus reflecting directly its intrinsic nature. In peer-to-peer networks (cf. Sec. V), for instance, overlay management is implemented inherently by the P2P facility and can be considered a P2P system's inherent management capability.

3. Autonomous principle: Management is autonomous and does not involve any external technical intervention. This principle is also implied by the inherent principle and leads to the adoption of purely self-managing mechanisms. It is pure in the sense that *any* functional aspect is autonomous, including the enforcement of high-level business goals and physical intervention, such as the replacement of faulty devices.

4. 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 follow the autonomous principle.

Furthermore, INM defines the following principle that addresses the *evolutionary design* methodology:

5. Evolution principle: The architectural design principles 2-4 are to be implemented and shall be supported by technical developments in a way that they can be gradually adopted. This principle is essential in that it allows the accommodation of currently established approaches, the nondisruptive development of management functionality, and the accelerated adoption of higher degrees of inherence, autonomicity and abstraction by novel technological innovations.

While the architectural principles 2-4 are theoretic in nature, INM breaks down the evolutionary design into a three-dimensional functional design space that allows for a gradual adoption of these principles to various and practical degrees. Thereby, a three-dimensional disk is formed, which is shown in Fig. 1. In the center, INM designates the extreme case where principles 2-4 are adopted in their pure form.

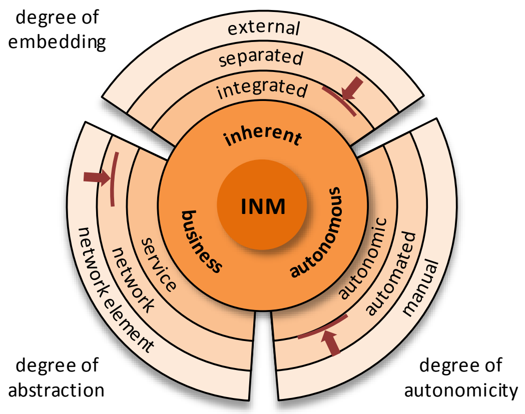


Fig. 1. INM evolutionary disk: degree of embedding (top), degree of autonomicity (right), and TMN functional hierarchy (left) according to [16].

On the axis of the **degree of embedding**, INM provides scope for a relaxation of the inherent principle. Management processes can be implemented either as external, separated, integrated, or inherent management capabilities of the network. Integrated is weaker in that instead of indistinguishable management functionality, it designates well identifiable management capabilities that are modular and visible, but 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, today's weakly distributed management approaches (e.g. RMON). External management processes include traditional management paradigms widely used today (e.g. SNMP).

On the axis of the **degree of autonomicity**, INM allows for different degrees of autonomous management, from manual to fully autonomous processes. Manual refers to the direct 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 behavior.

On the axis of the **degree of abstraction**, different levels of management according to the TMN functional hierarchy [16] 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*.

An essential philosophy is that INM does not *force* the adoption of the extreme case, or any specific degree on any of the functional dimensions. Instead, different parts of the network may adopt their specific degree of embedding, autonomicity, and abstraction, based on practicability and domain- or application-specific goals and requirements. At the same time, INM proactively supports evolution in the functional dimension in a technological aspect. If design issues are considered at the design time of new components, then newly introduced components may encapsulate existing management functionality in a way that allows for a nondisruptive transition to a purer INM system.

IV. Architectural Elements

The architectural elements proposed by INM are based on the principles and functional dimensions described in Sec. III. Let us first consider the high-level architecture of INM, shown in Fig. 2. The figure depicts a practical case of INM that relaxes the pure paradigm on each of the functional dimensions. In the degree of embedding, we assume that management functions are closely tied to the service processes, preferably in an inherent or integrated manner (cf. Fig. 1). Consequently, INM processes closely collaborate with the service processes in which they are embedded. In the degree of autonomicity, we observe that a certain set of management tasks cannot be automated practically and therefore must remain external,

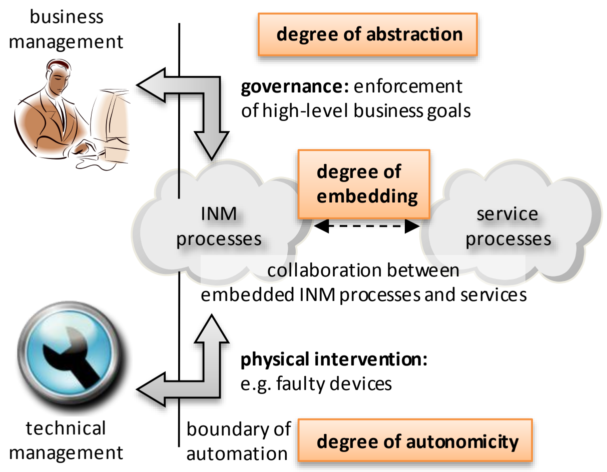


Fig. 2. Traditional (left) and in-network management (right).

indicated by the boundary of automation. On the left side of this boundary, both technical management, including physical intervention, and business management, remain. In the latter, a high level of abstraction will likely dominate, where business goals govern the network behavior. We next drill down the high-level architecture into two fundamental elements:

Functional components (FCs) encapsulate network service and management functionality within a single element. We use the term service in a fairly broad sense, including network services and functionality. This makes the concept of FCs applicable to any layer of the TMN functional hierarchy. Each functional component implements a set of FC properties, which impose a well-defined yet flexible set of characteristics on FCs and management structures. They are essential in the process of assembling complex management functions from simple ones in close interrelation with services. As such, functional components are vital in coping with the complexity of large-scale network management.

Management capabilities (MCs) are fine-grained entities that implement specific management functionality or parts thereof. Individual MCs may be composed to create more complex management processes from simple ones, within the same or across multiple FCs. Depending on the degree of embedding, MCs may be external, separated, integrated or inherent with respect to a specific service process.
Fig. 3 shows how FCs map into the high-level architecture and their relation to MCs. Observe the distinction of FCs into self-

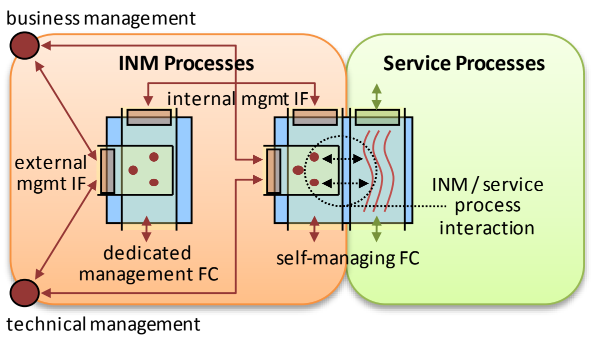


Fig. 3. INM and service processes.

managing FCs, which mediate between INM and service processes in terms of management, and dedicated management FCs, which contain only management-specific functionality. Furthermore, FCs interact with business and technical management and among each other in order to collaborate in performing management via external and internal management interfaces, respectively. Fig. 3 also shows how FCs embed management capabilities according to the inherent (wiggly lines) and integrated (circles) degree of embedding.

A. *Functional Components*

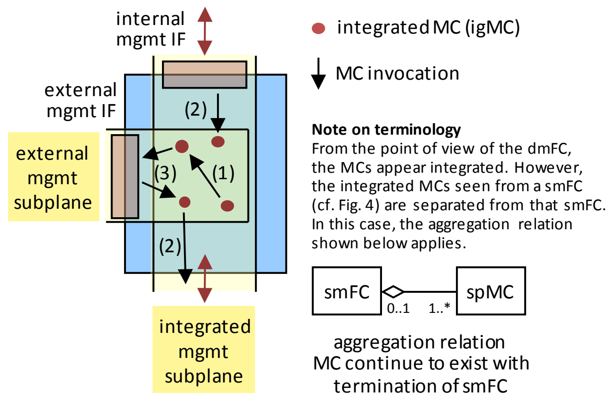
Functional components (FCs) are the basic elements in a communication network that can encompass both management and service functionality in one entity. An FC might represent, for instance, a protocol (sub)layer (e.g. a TCP/IP module or a MAC sublayer) or any other software module that encapsulates a specific service. FCs are distinguished into two types, termed *self-managing* (smFC, Fig. 4) and *dedicated management FCs* (dmFC, Fig. 5). 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 neighbor table).

Let us first consider the self-managing FC, shown in Fig. 4, which offers its service via the service interface (e.g. the sending of frames in a MAC module). An smFC provides two additional interfaces that enable it to communicate with either external components or other FCs for the purpose of management. The internal management interface is for any collaboration between FCs in order to access one another's MCs so distributed management objectives can be achieved collaboratively. The external management interface is related to governance and mediates between external (business and technical management according to Fig. 3) and internal (both integrated and inherent) management.

Motivated by the separation into several distinct degrees of embedding, the smFC explicitly reflects this distinction in that the smFC's management capabilities are arranged logically into three management subplanes, as shown in Fig. 4. Note that this does not imply any functional distinction beforehand. However, the rationale is to make explicit the migration of management functionality towards higher degrees of embedding and to support it. We will detail on this aspect in conjunction with management capabilities in Sec. IV.B.

In Fig. 5 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. Due to the fact that a dmFC's management capabilities may be reused by several smFCs, this type of FC contains only integrated management capabilities which are published via the internal or external management interface.

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



Note on terminology
From the point of view of the dmFC, the MCs appear integrated. However, the integrated MCs seen from a smFC (cf. Fig. 4) are separated from that smFC. In this case, the aggregation relation shown below applies.

Fig. 4. Self-managing functional component (smFC).

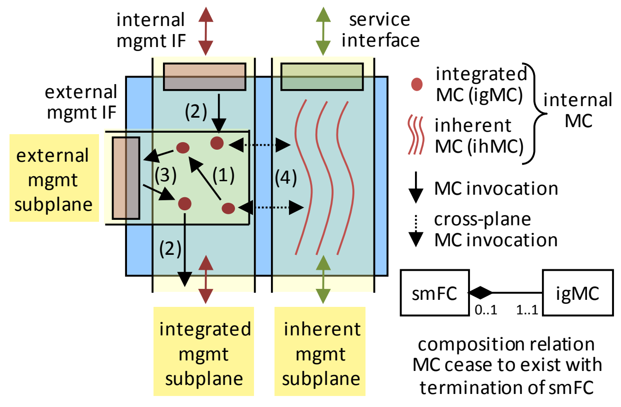


Fig. 5. Dedicated management functional component (dmFC).

from external systems (e.g. management stations) closer to the relevant self-managing FC. Fig. 4 and 5 include also a view on the distinction between integrated and separated management in a UML style. In this view, integrated MCs (igMC) of an smFC can be considered to follow a composition relation (Fig. 4), whereas MCs that are separated from an smFC (abbreviated spMCs) and contained within a dmFC match an aggregation relation (Fig. 5).

In order for FCs to be combined into more complex management processes, they are supported by a set of key FC properties. These properties provide the abstraction of a well-defined management process that has a well-known set of capabilities and with which well-defined interactions are possible. On one side, FC properties describe the mechanics of how FCs and their embedded management capabilities are integrated into more complex management processes. On the other side, they specify characteristics that allows the FC to be governed and observed by external entities. In order for a consistent overall management system that is composed of a multitude of functional components, and to guarantee that the overall management system is able to achieve high-level goals consistently, the implementation of all of the following properties is mandatory for each functional component:

Self-descriptive property: Any FC describes the service and management functionality it provides in terms of its interfaces and implemented MCs (cf. Sec. IV.B). Furthermore, service and management descriptions may specify dependencies which indicate that the collaboration with other FCs is required. The self-descriptive property allows any management system to discover and access FCs and FCs to discover one another. For example, semantic descriptions could be helpful to implement the self-descriptive property.

Composability property: In order to be able to create new services based on existing ones, FCs should be able to assemble for producing composite services. When different FCs are combined, they also bring together their internal management capabilities for potential interaction between each other. The FCs should have a standard set of interfaces so that they can be composed to produce composite services. The self-descriptive property is a prerequisite to the automation of composition, but might not be needed for e.g. statically or manually composed services and management.

Auditability property: When machines are allowed to control themselves, there is a natural risk of instability. Even unlikely situations may still occur and cannot be completely accounted for beforehand. However, the stability of communications systems is of paramount importance and FCs need to be equipped with very robust management control loops. Furthermore, FCs need to take into account the states (faults and performance) of other FCs on which they rely and which rely on them. For that purpose, functional components must support, whenever necessary, the performing of appropriate audits. Such audits may include (1) the tracing of self-management tasks carried out by the FCs, (2) reasoning about performing particular self-management tasks (e.g. for diagnosis), (3) configuration integrity checks, including software version control and patch details, and (4) accountability (integrity and fulfilment of agreements).

Governance property: Each FC is owned by one or more organizations or persons. Each FC can be governed by business service management (e.g. for service creation), technical administrators (e.g. for physical capacity increase or component exchange), or through the vendor (e.g. for software bug-fixes and upgrades). Plus, depending on the scenario, ownership and governance can be carried out by a single party or a number of separate parties. The INM architecture supports governance domains to reflect this real world organizational structure and for conflict resolutions.

B. Management Capabilities

Management capabilities are the fine-granular elements from which more complex management functions are constructed (e.g. performance monitoring, situation awareness). They can reside at any degrees of embedding as introduced in Sec. III, Fig. 1. While separated, integrated, and inherent management capabilities always reside inside of FCs, external management capabilities are located external to any FC. Specifically, inherent MCs are closely tied to the service which is provided by its encapsulating FC. Normally this capability will not be visible outside the FC. In Fig. 4 and 5, this type of capability is indicated by the wiggly line. Integrated MCs are capabilities that reside within an FC and have a definite relationship with the FC's provided service, but which are not generic enough for use by other FCs. Separated MCs always reside in a dmFCs and are generic in that they may be used by a number of other FCs. Finally, external MCs represent entry points for business and technical management as displayed to the left of the boundary of automation in Fig. 2.

Management capabilities should be designed such that they can potentially run at any level of embedding. What level of embedding an MC resides in is at the discretion of a functional component developer. The amount of exposure they wish to give a capability and how service-related the capability is will determine the level of embedding. This approach lends itself to the realization of the evolution principle described in Sec. III. An integrated management capability can be pushed down to a lower level of embedding and become inherent. This also results in the potential to create an external library of MCs which could be queried and used by FC developers.

Integrated MCs are of specific interest because they allow the incorporation of management functions in a flexible and modular way. They have a number of properties which define them. They have the ability to **communicate** with each other, within the same FC and also between different FCs. They have a **self-descriptive** mechanism that acts as a feeder to the self-descriptive FC property of the FC which they reside in. Management capabilities must also be **discoverable**, which may be mediated via its hosting functional component.

Besides service-specific integrated MCs, it is possibly to dynamically add MCs to an FC at runtime. As such, they provide the space for any additional network management task that is not provided inherently by the smFC. For example, a monitoring function that monitors the state of a TCP/IP module relevant to performance management is typically located within the scope of integrated MCs. In contrast to inherent MCs, integrated management capabilities are well-distinguishable management structures, or (part of) network management functions. They possess their own interface that in turn can be published via an FC's management interfaces.

Furthermore, a large part of the communication related to the execution of more generic management functions takes place at the level of individual integrated MCs within the integrated management plane. In Fig. 4 and 5, for instance, an integrated MC is invoking another integrated MC. This interaction could be used, for instance, by an event mechanisms to propagate failure information between different network functions. In the same figures, interaction (2) occurs between integrated MCs of different FCs, mediated through the internal management interface by each FC.

Two additional interactions are of specific interest due to their mediating between the external/integrated and integrated/inherent management planes. In Fig. 4 and 5, interaction (3) designates communication between an external management component and an integrated MC. The integrated management capability is thus invoked by an external entity and vice versa via the FCs external management interface. This kind of invocation is typical for the enforcement of business objectives to an FC's internal management functionality. Interaction (4) occurs between the integrated and inherent management planes, also possibly in both directions. In one direction, an MC could provide access to inherent management functionality from the integrated management plane. For example, a P2P preferences capability could allow the integrated management to access preferences of the P2P facility through a well-defined preferences capability interface. In the other direction, an MC may allow the inherent management plane to invoke specific functions that it cannot handle alone. In the P2P example, this might be a security capability that handles authentication in cases where the P2P FC cannot handle security issues by itself.

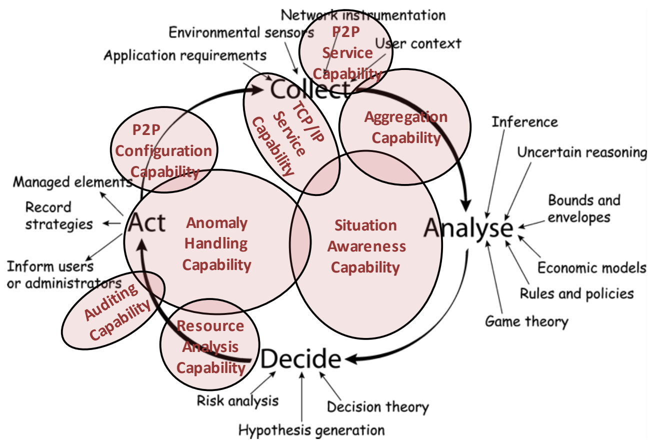


Fig. 6. Control loop with management capabilities (based on [17]).

While in this paper we focus on the architectural principles and elements, we emphasize the concreteness of our proposal in Fig. 6. In the figure, several types of management capabilities are placed inside the control loop that is taken from [17]. This example illustrates the kind of management capabilities that are currently developed within the scope of the 4WARD project [2]. In the figure, each of the capabilities are mapped to a set of distributed FCs. Overlap between capabilities indicates communication between MCs, either within or beyond single FCs. Furthermore, each of the capabilities may be realized at any degree of embedding.

V. Application of INM To Peer-to-Peer Systems

Given the above described INM paradigm and architecture, one question comes to the fore how one should design for such a framework. This is important for two cases, (1) the design issues when applying the INM system to existing network or service management and (2) the design of a future Internet architecture from scratch in a clean slate manner.

In existing systems, we cannot really add more management functions than there are already. But in many cases existing built-in (self-)management functions need to be accessed and adapted. Also management functions currently located in a central spot can be adapted to a more decentralized design and located closer to the network functionality. For these functions, INM provides the necessary flexibility to build a suitable management capability model and integrate each function at one of the degrees of embedding.

In case of the future Internet, the INM features can be much better exploited than for existing systems. The design approach in an extreme version of the future Internet is to build as much management functionality as possible directly and inherently into the future Internet functions and protocols. Based on that, the set of management capabilities within the functional components should cover the whole management space, and the governance should be designed at a degree of abstraction that deals with business level issues only.

Since we do not have a future Internet design yet, we have chosen to apply INM to a P2P networking example. P2P is interesting because of its ubiquity in today's Internet and because it has quite a number of management functions built into the system itself. In the following we show how the actual INM system would work in a P2P data management scenario (Fig. 7). According to this scenario, P2P systems will provide a uniform mechanism for accessing content organized as information elements, e.g. by means of structured DHTs. Given an identifier for an element, a P2P infrastructure will decide what the optimal sources are for retrieving the content. Different algorithms and strategies for content distribution may be used (depending on the specific P2P implementation) together with techniques for content adaptation to ensure a delivery that is optimally adapted to a particular user as well as network resources and policies. We assume that the P2P machinery establishes a dynamic communication overlay on top of the underlying network infrastructure for both internal management and data transfer purposes. The management capabilities (cf. Fig. 7), comprise the following functions: neighbor discovery for topology building, choosing the optimal delivery method, access control and enforcement, error management, and ICMP error reporting.

The topology building MC allows bootstrapping and maintaining the relationships between parts of the P2P machinery. The P2P system may contain a set of functions for neighbor discovery and topology building based on a set of objectives in terms of communication strategy and neighbor relationships (e.g. gossiping protocols or beacons). The system designers could simply choose either one of the strategies and type of neighbor relationships that are most suitable for them. Alternatively, an INM capability co-located with the neighbor discovery capability might make the decision, based on information received via the external management interface or based on measurements performed automatically through the interaction with TCP/IP. The neighbor discovery MC can access details regarding the physical connectivity between neighbors, such as link segment types, one-way transmission delays, and live data related to network monitoring.

The monitoring and maintenance capabilities in the topology discovery capability do not perform the monitoring task itself, but rather read from the P2P system the results and make those accessible externally to other FC's management capabilities. For example, a user might want to supervise the system through a management GUI or switch to a different P2P system. Also the management capability might export the

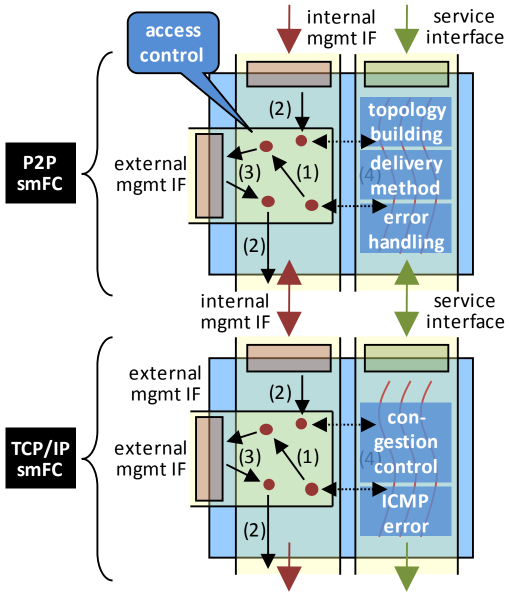


Fig. 7. Example: in-network management for P2P systems.

monitoring results to an FC, which is run by the P2P system developer to get feedback on the users' problems.

The algorithm in the P2P machinery for choosing the best delivery method for an information object might be designed as yet another MC. We assume that the algorithm takes into account a set of networking characteristics, which the P2P system figures out by itself. However, since it is only able to measure between peers and does not know the network conditions in between, those measurements might be incorrect or not desired by an ISP. So an INM management capability integrated within this FC may interact with an ISP's network management capability to receive network conditions to improve or ease the decision process. With integrated MCs, the algorithm could take advantage of information available through FCs in the INM architecture, including, but not limited to, the following: (1) identify a set of data caches relevant to the destination, based on the topology information of the network; (2) instant (or historical) values about using a certain destination to load the object from, but measure by other peer for other objects; (3) determine whether the SLA of the destination allows for P2P delivery at all.

The access control and enforcement functionality allows for the P2P machinery to restrict the access to the objects based on user credentials and roles. The INM framework could provide the following functionality that would help in this mission: (1) a dissemination protocol with automated updating of the in-network access control configuration mechanisms and (2) an automatic enforcement of access policies via a two-way API for communication between the P2P machinery and the INM framework that controls the network-attached devices where the objects reside physically.

The troubleshooting and diagnostic capabilities embedded in the INM framework allow for a detailed analysis of network-related problems. For example, in the event where an information object cannot be delivered to a destination, the

P2P machinery may take advantage of integrated management capabilities in order to determine the cause of the error and to decide whether this is a transient or permanent problem. As such, INM could determine whether a large frame loss noticed by the P2P machinery is the result of temporary congestion or of a severe failure in one of the physical links. In the above example, P2P would make use of the management capabilities made available by the network, which is concerned about path information. The evaluation of edge-to-edge delay is a typical component that could be part of INM. In combination with a similar service implemented at the P2P level, it may allow to determine whether a particular problem was localized in the end nodes or within the network.

VI. Conclusion

The future Internet will bring many challenges with it and its management will be one of the most taxing. Techniques to enable its management be efficient and transparent will be pivotal. This paper proposes five architectural principles for in-network management, a new paradigm for the management of future communication networks. Based on this set of clean slate principles and an evolutionary design space, we proposed architectural elements that act as enabling building blocks to embed management capabilities inside the network. We showed by the example of P2P-based data management how these architectural elements can be practically applied, in an evolutionary way, to real world networking scenarios.

We believe that the proposed concepts are an essential step in sustaining the manageability of future networks and in the feasibility to gradually implement the necessary changes to allow for noncomplicated and scalable management. Moreover, the adoption of a higher degrees in the functional dimensions of in-network management will lead to significant long-term benefits in terms of capital and operational expenses due to the increased automation and more abstract specification of management processes and objectives, respectively. While the proposed concepts embrace a large spectrum of today's complex and heterogeneous management systems, we can see that they will also stimulate the design of networks and services to support the development towards a clean slate approach in a controlled and rapid way.

Future work includes the refinement of details in the architectural elements and their interaction and composition to form more complex management functions. While we have gained first experiences with a framework for in-network management and its application to bio-inspired networking [18], in-network support for running embedded management processes is to be extended along the axis of functionality and performance. Specifically, we are working on the detailing of a management capability model and the creation of a library of efficient management functions for being integrated into the management plane. Finally, we are looking at how security issues can be achieved more inherently based on the abstractions provided by in-network management.

Acknowledgment

This work is supported in part by the European Union through the 4WARD project (http://www.4ward-project.eu/) in the 7th Framework Programme. The views expressed in this paper are solely those of the authors and do not necessarily represent the views of their employers, the 4WARD project, or the Commission of the European Union.

References

[1] M. Johnsson, J. Huusko, T. Frantti, F.-U. Andersen, T.-M.-T. Nguyen, and M. P. de Leon, "Towards a new architectural framework – the Nth stratum concept," *Proc. MobiMedia'08*, Oulu, Finland, Jul. 2008.

[2] N. Niebert, S. Baucke, I. El-Khayat, M. Johnsson, B. Ohlman, H. Abramowicz, K. Wuenstel, H. Woesner, J. Quittek, and L. M. Correia, "The way 4WARD to the creation of a future Internet," *Proc. PIMRC'08*, Cannes, France, Sep. 2008. To appear.

[3] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, " A clean slate 4D approach to network control and management," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 41-54, Oct. 2005.

[4] C. Dovrolis, "What would Darwin think about clean-slate architectures?" in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 1, pp. 29-34, Jan. 2008.

[5] H. Ballani and P. Francis, "CONMan: Taking the complexity out of network management," *Proc. of the 2006 SIGCOMM Workshop on Internet Network Management*, Pisa, Italy, Sep. 2006, pp. 47-51.

[6] T. S. E. Ng and H. Yan, "Towards a framework for network control composition," *Proc. of the 2006 SIGCOMM Workshop on Internet Network Management*, Pisa, Italy, Sep. 2006, pp. 47-51.

[7] D. D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, "A knowledge plane for the Internet," *Proc. SIGCOMM'03*, Karlsruhe, Germany, Aug. 2003, pp. 3-10.

[8] M. Wawrzoniak, L. Peterson, and T. Roscoe, "Sophia: An information plane for networked systems," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 1, pp. 15-20, Jan. 2004.

[9] T. Bullot, R. Khatoun, L. Hugues, D. Gaïti, and L. Merghem-Boulahia, "A situatedness-based knowledge plane for autonomic networking," *International Journal of Network Management*, vol. 18, no. 2, pp. 171-193, Mar. 2008.

[10] L. B. Ruiz, J. M. Nogueira, and A. A. F. Loureiro, "MANNA: A management architecture for wireless sensor networks," *IEEE Communications Magazine*, vol. 41, no. 2, pp. 116-125, Feb. 2003.

[11] 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, Oct. 2007.

[12] Y. Cheng, R. Farha, M. S. Kom, A. Leon-Garcia, and J. W.-K. Hong, "A generic architecture for autonomic service and network management," *Computer Communications*, vol. 29, no. 18, pp. 3691-3709, Nov. 2006.

[13] B. Ahlgren, L. Eggert, B. Ohlman, and A. Schieder, "Ambient networks: Bridging heterogeneous network domains," *Proc. PIMRC'05*, vol. 2, Berlin, Germany, Sep. 2005, pp. 937-941.

[14] M. Zach, D. Parker, C. Fahy, R. Carroll, E. Lehtihet, N. Georgalas, R. Marin, J. Serrat, and J. Nielsen, "Towards a framework for network management applications based on peer-to-peer paradigms," *Proc. NOMS'06*, Vancouver, Canada, Apr. 2006.

[15] C. Jelger, C. Tschudin, S. Schmid, and G. Leduc, "Basic abstractions for an autonomic network architecture," *Proc. AOC'07*, Helsinki, Finland, Jun. 2007.

[16] 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.

[17] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, f. Saffre, N. Schmidt, and F. Zambonell, "A survey on autonomic communications," ACM Transactions on Autonomous and Adaptive Systems, vol. 1, no. 2, pp. 223-259, Dec. 2006.

[18] C. Foley, S. Balasubramaniam, E. Power, M. P. de Leon, D. Botvich, D. Dudkowski, G. Nunzi and C. Mingardi, "A framework for in-network management in heterogeneous future communication networks," *Proc. MACE'08*, Samos Island, Greece, Sep. 2008. To appear.