Chapter 2
A System Overview

Martin Johnsson

Abstract The 4WARD System Model is described, defining the structure and behavior of a communication system that is to be constructed as well as its generativity, i.e., how bigger and more complex future systems and networks can be built by using a small set of generic concepts. It presents the project four tenets. Then, an Architecture Framework is shown, providing a unified component-based design process, which defines a seamless step-wise though iterative process for deriving a software-based network architecture using as input a set of technical requirements. The Architecture Pillars, described in detail, are: In-Network Domain Management, Network of Information, Generic Path, and the Physical Virtualized Substrate. The Architecture Framework is presented in terms of Strata, Netlets, and the Design Repository. The Design Process is also addressed.

2.1 Background and Motivation

This section describes the 4WARD System Model, which defines the structure and behavior of a communication system that is to be constructed as well as its generativity, i.e., how bigger and more complex future systems and networks could be built by using a small set of generic concepts.

Through 4WARD, a new approach to networking based on the analysis of both the success factors of the Internet (seen as the core Internet design principles and core IP protocols) as well as the factors that have led to ossification and the patchwork type of the IP evolution of recent years has been developed.

The Network of the Future must be based on a new set of *Internetworking principles*. These principles are characterized below as four programmatic tenets:

M. Johnsson (✉)
Ericsson Research, Stockholm, Sweden

L.M. Correia et al. (eds.), *Architecture and Design for the Future Internet*,
Signals and Communication Technology,
DOI 10.1007/978-90-481-9346-2_2,

1. Let 1000 Networks Bloom
   We will explore a new approach to a multitude of networks: the best network for each task, each device, each customer, and each technology. Unlike the multitude we had in the past, where different incompatible technologies were competing with each other, we want to create a framework that will allow these many networks to bloom as a family of interoperable networks coexisting and complementing each other.
2. Let Networks Manage Themselves
   The main limits of current technologies are the scaling up to very large network sizes, and the needed human intervention which is associated with considerable cost, errors and with an inherent slowness in reacting to changing network conditions. What we would like to have is a management entity as an inseparable part of the network itself, generating extra value in terms of guaranteed performance in a cost effective way, and capable of adjusting itself to different network sizes, configuration, and external conditions.
3. Let a Network Path Be an Active Unit
   We want to consider a path as an active part of the network that controls itself and provides customized transport services. An active path can provide resilience and fail-over, offer mobility, simultaneously use multiple different sequences of links, secure and compress transmitted data, and optimize its performance all by itself.
4. Let Networks Be Information-Centric
   Users are primarily interested in using services and accessing information, not in accessing nodes that host information or provide services. Consequently, we want to build a network as a network of information and services that may be mobile and distributed. In such a network, the users just accesses items of interest by their name while the data locations can be completely hidden.

These tenets, together with the understanding of the current situation of today's Internet, formed the main drivers for the definition of the 4WARD Technical Requirements [1], which laid the foundation for technical work within the 4WARD project. This work ultimately resulted in the 4WARD System Model, which is described in the following section.

2.2 The 4WARD System Model

Figure 2.1 depicts the 4WARD System Model, which has been developed with the Tenets and the 4WARD Technical Requirements [1] as main principal input. The system model gives the necessary definitions, specifications, principles and guidelines for designing, building, deploying, and manage interoperable network architectures. For that purpose, the 4WARD System Model consists of an Architecture Framework and a set of Architecture Pillars which provides the essential technologies in many of the network architectures anticipated and required for the future networks, though it is possible to also deploy and use them in migration scenarios. With the 4WARD System Model we expect significant efficiency gains in the de-

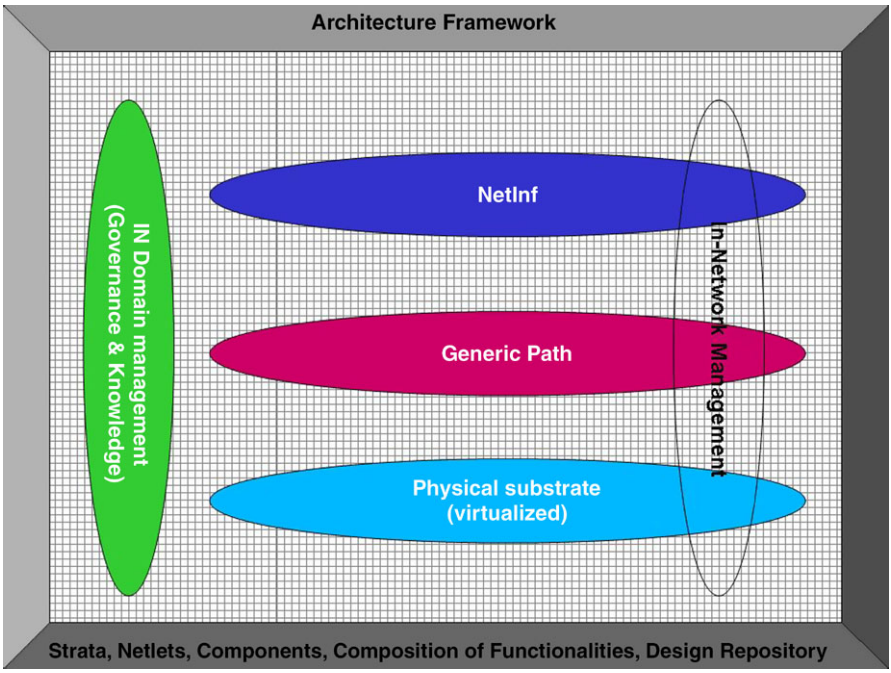

Fig. 2.1 The 4WARD System Model

sign, management and operation of networks, which is one of the key challenges in both current and future networks. The Architecture Pillars have been defined using a new set of concepts and technologies to address emerging business models and new types of applications:

- A new abstraction and model of the physical and virtualized infrastructure, including all of transmission, processing, and storage resources.
- ONE modular and extensible connectivity concept, supporting all modes and topologies of endpoint associations.
- A new open model and API for content and information management. Search and retrieval of information objects using a persistent identity.
- Management providing an inherent capability of the functions in the network.

The Architecture Framework provides a unified component-based design process, which defines a seamless step-wise though iterative process for deriving a software-based network architecture using as input a set of technical requirements. The design process includes the architectural principles and re-usable design patterns at various levels of abstractions out of which families of interoperable network architectures can be defined.

The Architecture Pillars: In-Network Domain Management, NetInf, Generic Path, and the Physical but virtualized substrate (and each of them in turn define their own respective frameworks or architectures) themselves to be defined by using the Architecture Framework.

The Physical Substrate provides an abstraction of the physical resources of any network spanning from the smallest to the largest. The abstraction is the key for coherent virtualization and management of those underlying resources across domain borders. The result of a virtualization operation is a virtualized network, providing resources onto which an operator is free to instantiate its own choice of functions, protocols, etc., for example Generic Paths and NetInf.

The Generic Path provides a generalized transport mechanism to transfer data between entities in the network. The recursive Generic Path concept is able to model virtually any type and level of transport, be it point-to-point or multipoint-to-multipoint, or supporting transport on links at the physical level, or end-to-end across networks. Generic Paths specifically give support for the dissemination of information objects.

NetInf (Network of Information) provides for identification, management, and dissemination of information objects. NetInf is a new abstraction of information (and service) management, where applications do not need to be aware of where an information object is stored.

In-Network Management (INM) is omni-present in all network functionalities. It provides design patterns and interfaces as well as more specific mechanisms, facilitating various degrees of self-management capabilities. This spans from such capabilities living 'beside' the functionality it is supposed to manage, and then all through to functionalities being fully and inherently self-managed.

A special case of In-Network Management is In-Network Domain Management, which provides self-management capabilities on domain as well as inter-domain scale. The Knowledge function (also known as Knowledge stratum) discovers, gathers and further infers status of network topologies, resource and context status by querying the network functionalities operating in the network, for example Generic Paths and NetInf. The Governance function (also known as Governance stratum) provides control and management of network functionalities, and governs by querying the status of the network from the Knowledge function. The Governance function will decide out from policies (provided by a network administrator) as well as the network status what network functionalities shall operate in the network. Governance and Knowledge functions are also instrumental for the interconnection and composition of networks and domains, where dynamic and highly automized creation of SLAs is supported.

The following sections provide an overview and introduction of the concepts and technologies that make up the foundation of the Architecture Pillars, and it serves as an introduction to the contents provided through Chap. 4–10.

2.3 The Architecture Framework

2.3.1 Strata, Netlets, and the Design Repository

The Architecture Framework must provide ways to (i) guide the Network Architect to allocate the required network functionalities and (ii) assure the interoperability within families of network architectures.

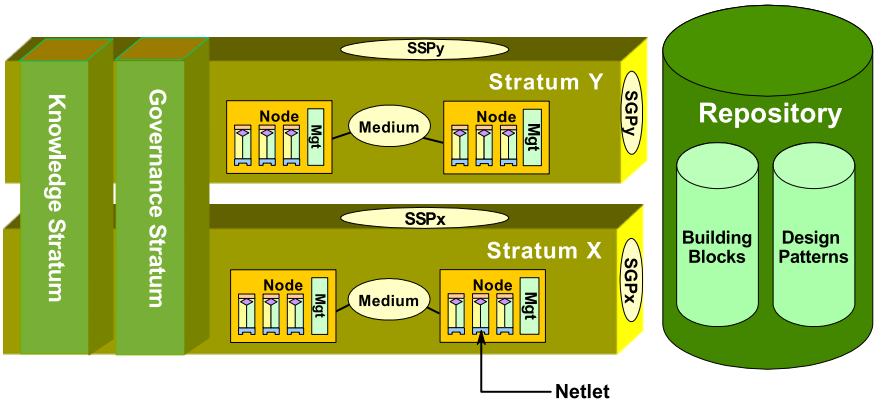

Fig. 2.2 High-level view of 4WARD Architecture Framework

As can be seen in Fig. 2.2, the following main components constitute this framework (see Chap. 4 for further detail):

- A Stratum is modelled as a set of logical Nodes which are connected through a Medium that provides the means for communication between the Nodes inside this stratum. This stratum encapsulates functions that are distributed over the nodes. These functions are provided to other strata through two well known interfaces (that can be also distributed over the nodes): The SSP (Service Stratum Point) that provides the services to the other strata located on top of the respective Stratum and to the vertical strata. Figure 2.2 shows Stratum Y using the services provided by Stratum X through SSP_X. The SGP (Service Gateway Point) offers peering relations to other strata of the same type.
- Strata can manage themselves. For example, when a routing service stratum is deployed, it organizes itself onto the physical infrastructure. The deployment will be in accordance with the specification of the logical nodes and the medium of the stratum, taking then into account the topology, capabilities, and resource status of the nodes and links in the physical infrastructure.
- Horizontally stacked strata (as shown in the middle of Fig. 2.2) are related to the transport and management of data across networks. Within such strata, Netlets can be considered as containers for networking services. They consist of functions/protocols inside a Node that are needed to provide the services. By virtue of containing protocols, Netlets can provide the Medium for different Strata, i.e. inside the same Netlet there could be functionalities that are related to different strata. Figure 2.2 shows such Netlets implementing media for different strata inside the same node.
- The two vertically oriented strata provide Governance and Knowledge for an entire network (i.e. a set of horizontal strata). The Knowledge Stratum provides and maintains a topology database as well as context and resource allocation status as reported by a horizontal stratum. The Governance Stratum uses this information, together with input provided via policies, to continuously determine an optimal

configuration of horizontal strata to meet the performance criteria for a network. The Governance Stratum also establishes and maintains relations and agreements with other networks.

The Repository contains the set of Building Blocks and Design Patterns for the composition of functionalities (i.e., to construct the strata and the netlets) for specific network architectures, including best practices and constraints to ensure interoperability between network architectures.

2.3.2 The Design Process

Evolution of today's networks including the Internet suffers from the inability to be extended in a consistent and reliable way while maintaining certain assured properties, such as security, quality of service, reliability even in the broader context. Much effort has to be spent for standardization, development and regression testing when introducing even minor feature improvements, before deploying them on a network-wide basis. Upgrading of a large installed base of network elements means a big technological challenge and financial risk to the network operator and service provider.

4WARD has succeeded in setting up a design process that in the future will enable new network designs to be developed, tested and deployed without impacting the installed network basis, when based on this 4WARD architecture framework and building upon the recent progress in network virtualization. The innovative 4WARD network design process leverages advantages of model-driven software engineering techniques and the experiences in design and composition of web services, based on OSGI principles [2].

As shown in Fig. 2.3, the following phases are considered in the design process:

1. **Requirements Analysis:** Starting from the business idea and requirements, the goal of this step is to decompose them into the high level functionalities that should be realized by the architecture to be designed. The output of this phase is mainly the identification of the macroscopic architectural view of Strata, a first draft of the main network components, and the specification of technical requirements for further refinement of the architecture.
2. **Abstract Service Design:** During this phase, the technical requirements and the high level functionalities derived from these will be turned into abstract functionalities and ways how they can be composed, following generic principles and design patterns. The result of this design phase is the specification of the Netlets operating at node level, and the Strata that constitute the distribution of functionalities across the network nodes.
3. The **Component Design Phase** focuses on the detailed specification and composition of the Functional Blocks (FBs) used to implement the specific functionality. This includes the specification of the interfaces, properties, and requirements/prerequisites of the FBs. The output of this phase is the detailed design of

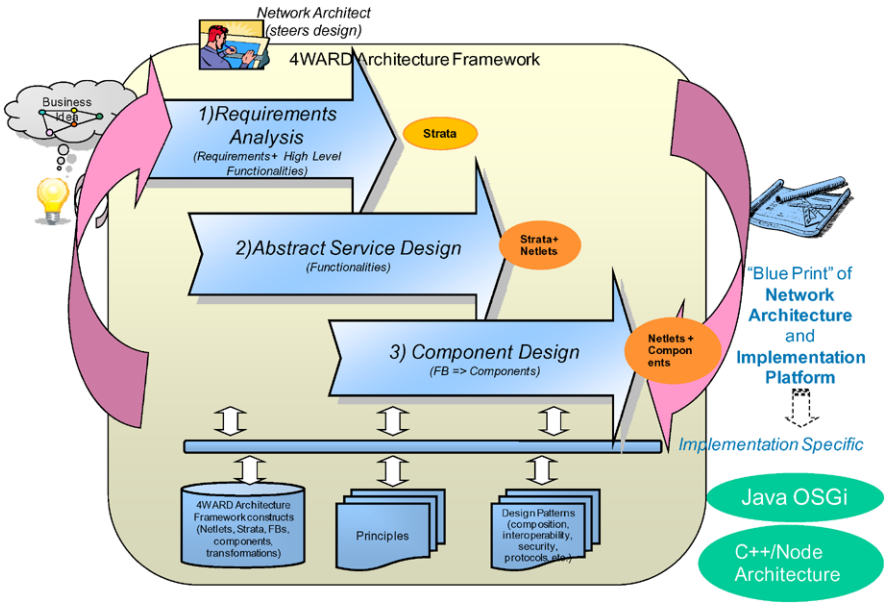

Fig. 2.3 High-level view of the 4WARD Design Process

the Netlets and software Components, which finally constitutes an "architectural blueprint" ready for instantiation on a network virtualization platform.

The entire design process is supported by an integrated design environment, which easily supports backtracking in iterative loops to redesign and improve the results of previous phases. In order to increase the reuse of architectural constructs and store the expertise and knowledge of the designing architect, an "architectural design repository" is used, which contains pre-built architectural constructs (abstract strata, netlets, components, functional blocks) as well as their derived instantiations, proven architectural design patterns on service and network composition, interoperability, security, etc.

2.4 In-Network Management

INM specifies two key architectural elements in order to realize distributed management within and across the network nodes: ***Management Capabilities*** (MC) and ***Self Managing Entities*** (SE). The MCs are encapsulations of management logic. The SEs are associated with a specific service and include relevant MCs for management of the service. Both elements are central to achieve autonomous behavior.

As part of the INM solution and design, algorithms have been developed for real-time monitoring, anomaly detection, situation awareness, and self-adaptation

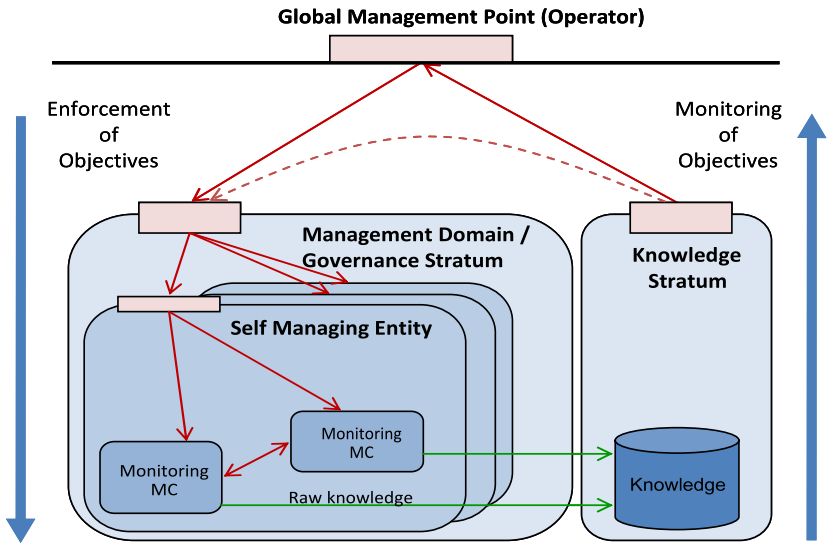

Fig. 2.4 INM relationship with Governance/Knowledge

schemes. The MC architectural element is the enabler of these algorithms. These algorithms provide best of breed mechanisms and patterns to address management tasks. They become important building blocks when designing networks. The 4WARD design process as described above includes an 'architectural design repository' which houses design patterns and network type building blocks available to the architect of the future networks. From a management perspective the algorithms developed for INM are key components of this repository which the architect can deploy as the need arises.

The 'management by objective' approach of INM is intrinsic to governance of networks and knowledge generation inside networks of the future. Both governance and knowledge are modelled as strata in the 4WARD architectural framework. Figure 2.4 shows management objectives being pushed downwards through the governance stratum, into the SEs and eventually into multiple MCs which carry out the tasks in hand. The MCs in the figure could for example implement a monitoring algorithm. The output of the monitoring algorithm is in essence unprocessed data. This is fed into the knowledge stratum and reasoned upon and more high level knowledge generated. This knowledge is then used, possibly fed back into governance if some modifications or tweaking are necessary or displayed at a higher level as feedback on the objectives which an operator applied to the network.

The algorithms developed and the management by objective approach which INM provides are key enablers in the realization of self managing, interoperable networks of the future.

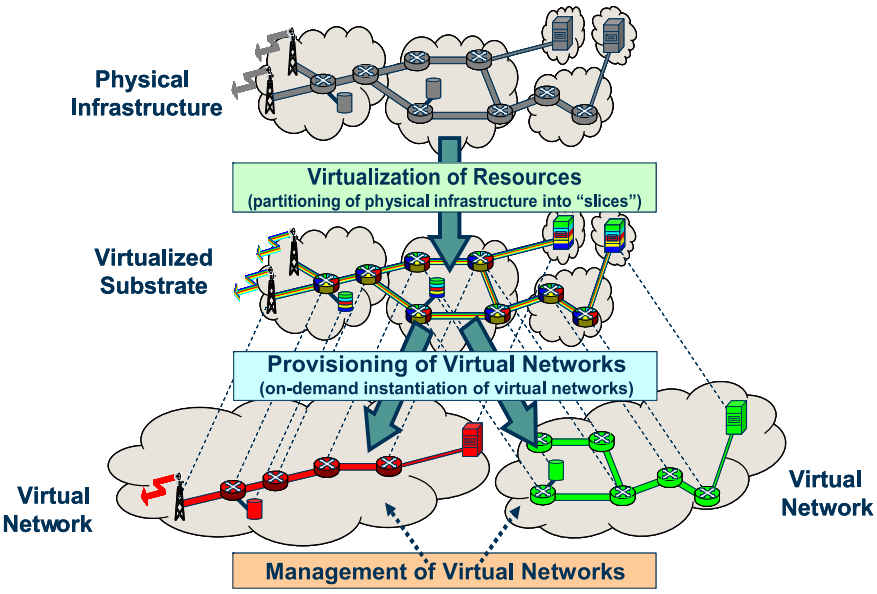

Fig. 2.5 Virtualization ecosystem

2.5 Network Virtualization

Virtualization has by now gained sufficient momentum as one of the key paradigms for future networking, as it has the potential to resolve the so-called "deployment stalemate" observed in today's Internet and foster the development of future networks paradigms. The straightforward use case for network virtualization is the scenario based on the decoupling of infrastructure ownership and virtual network operation.

The virtualization ecosystem encompasses three basic roles, namely (a) the infrastructure provider (having the capability to virtualize the physical infrastructure by partitioning them into 'slices'), (b) the virtual network provider (making the provisioning of complete end-to-end VNets by putting together 'slices' from the underlying infrastructure), and (c) the virtual network operator who is operating and managing a VNET. This is illustrated by Fig. 2.5. A service provider is then able to run specific services and applications on this VNet, which are then offered to end users.

Communication means between these actors and the definition of the respective interfaces constitute a cornerstone of the network virtualization architecture. This requires the specification of a formal virtual network description, allowing for flexibility, extensibility, scalability, interoperability and security. Since multiple business scenarios can be defined (ranging from vertical integration to a strict separation of roles), which imply different relationships of trust between them, the capability to define different levels of abstraction is also a key requirement. The 4WARD Resource Description Framework provides a language to describe virtual network re-

sources and topologies, including all possible constraints that might be applicable in each case. An object-oriented data model was defined with four basic classes describing specific network elements, namely nodes, links, interfaces, and paths.

4WARD network virtualization architecture breaks with the traditional clear separation between a "dumb" core and a feature-rich edge in service provider networks. In this scenario, scalability will be a major challenge, particularly in terms of provisioning, management and control of virtual networks. A framework and algorithms for scalable mapping and embedding of virtual resources into the infrastructure, including discovery, matching, and binding were developed. Initial results suggest that the efficient construction of virtual networks from shared infrastructure at large scale is indeed feasible.

One of the most important features of the current Internet, global reachability and inter-networking, will surely remain a requirement in the future. This means that virtual networks, which by definition are separated and isolated from each other, will still need to communicate, although in a more controlled way. A concept for facilities to provide interworking between virtual networks, the Folding Points, has been developed, including the basic elements (Folding Nodes and Folding Links), as well as mechanisms for deployment using the virtual network provisioning framework.

2.6 Generic Paths

New mechanisms for data transport face contradictory requirements: large flexibility vs. uniform interfaces to all transport entities and efficient reuse of functionality are required. This can be partially achieved by new protocols only in end systems, but in general, an approach how to structure protocols both at the edge and in the core, at various "layers" is needed. For example, network management needs to identify, inside the network, data flows of different types; they should be able to give account of themselves (e.g., about their desired data rate) and obey a common set of commands.

To support such requirements, we focus on the data flow and its path as a core abstraction, along with a design process for a variety of path/flow behaviors. This process can incorporate new networking ideas; examples are network coding, spatial diversity cooperation, or multi-layer routing and is suitable for both end system and in-network implementation; the deployment is supported by the Architecture Framework.

The starting point for the 4WARD transport architecture was to find (1) a development model that can support reuse and flexibility, (2) a proper execution environment within a node (end system or router) with naming and addressing structure and a resolution scheme, and (3) the core functions and APIs necessary for a path, as generic as possible. Together, this is the core of the Generic Path architecture. It approaches issue (1) by using an object-oriented approach to define types of Generic Paths and to structure their interfaces; issue (2) by defining a set of constructs (namely, entity, endpoint, mediation point, compartment, hooks, and

path) that describe the execution environment of instances of such path types; and issue (3) by selecting which operations should be possible on such paths (e.g., joining, splicing, or multiplexing). The concept shares some commonalities with OpenFlow, but concentrates on real-world necessities rather than on experimental usage; it also goes beyond merely modifying switching tables. To incorporate new networking ideas, all the relevant flows in a network share crucial commonalities and provide a common set of APIs with which to manipulate these flows. 4WARD's "Cooperation & Coding Framework" exploits such commonalities by addressing an entity that detects opportunities for turning on cooperation opportunities, like network coding, and can create the necessary path instances to setup a network coding butterfly. Mobility may be supported at different levels or compartments—and the realization of mobility at a session level is quite different from the realization of mobility at IP level, though they still share commonalities that can be defined through generalized mobility schemes. Thus, the GP framework allows the abstract description of a mobility process in terms of GP constructs, namely, entity, compartment, ports, path, and mediation point. Its realization can then resort to specific technologies adequate to the compartment we are considering in each case.

Based on this mindset, it becomes possible to develop powerful, custom-tailored path types. An example are path types for a Network of Information (described next), where the download of documents and the updating of location/caching tables can be tightly integrated and can access topology information to choose, for a document of interest, topologically close caches. Another example would be a path type to support the exchange of management information for In-Network Management entities, e.g., by compressing monitoring information more and more the further it is away from its source.

2.7 Network of Information

Today's networking is essentially about exchanging information between nodes. When accessing information, the request typically includes the host where the information shall be retrieved from, frequently in the form of a Uniform Resource Locator. This host-centric approach is often an obstacle for optimized transport of and easy access to information. Our approach to an information-centric architecture puts the information itself on the center stage. We take existing proposals that separate the host identity from the locator one step further by introducing information objects as first order elements in the network. In addition to classical scenarios such as content distribution, our work also encompasses scenarios that have so far not been discussed in the research community, e.g., the notion of real-world object tracking under the aegis of an information-centric architecture.

For the envisaged Network of Information (NetInf), we have developed an information model that constitutes a versatile and widely applicable framework for representing information in a wide sense. A clear split between the information itself and the location where it is stored is introduced. This eliminates the need for overloading locators and avoids putting them in the role of being an identifier and a

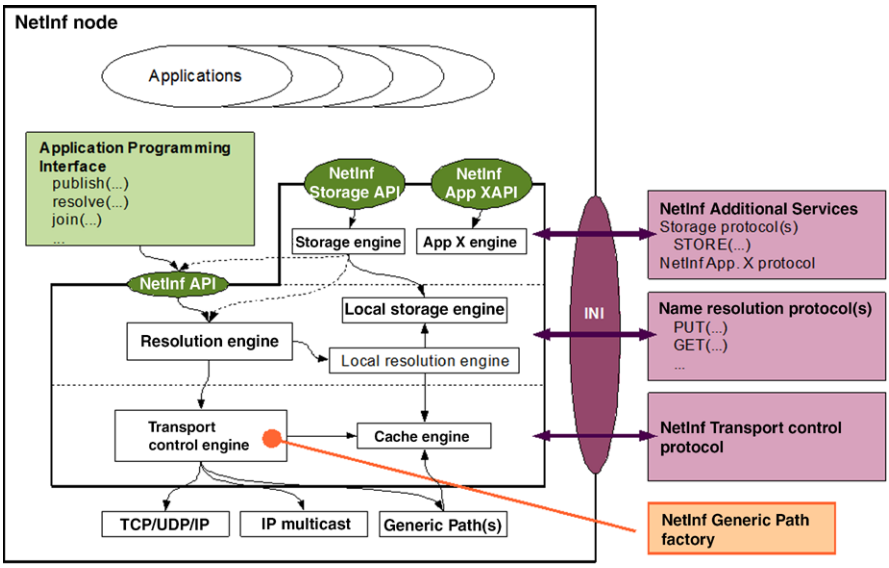

Fig. 2.6 NetInf high-level architecture

locator at the same time. The representation of the actual files containing the payload is called a *bit-level object* whereas the higher semantic level can be expressed by *information objects* that group or aggregate information.

The high-level architecture of a NetInf node is depicted in Fig. 2.6. The *NetInf Information Network Interface* (INI) at the right is the collection of NetInf protocols which are used to communicate to other NetInf nodes in the network. A uniform API exposed towards applications provides standard operations such as retrieving, publishing or updating information objects. This API can be extended with additional services. The Resolution Engine co-operates both with the local resolution engine when and if information objects can be found locally, but also with other remote resolution engines when such objects are stored elsewhere. Complementing the mobility schemes offered by the underlying transport, these mechanisms also provide a means to not only handle the mobility of nodes and networks, but also of information objects.

The NetInf Transport Control Engine is extremely flexible with regard to the transport mechanism that is utilized to transport the information objects or the requests. These transport mechanisms include, but do not mandate, the Generic Paths. Essentially, a set of adapted and optimized transport mechanisms applicable to information-centric networking are examples of specialized Generic Paths. The Transport Control Engine closely interacts with the Cache Engine which manages the caches that are used for short-term optimizations of data transport. The long-term memory of a NetInf system is provided by the (Local) Storage Engine. It uses the basic NetInf primitives to deliver and retrieve objects, while offering an advanced API that enables applications to manage the objects in the storage system, whether locally or remotely.

2.8 Conclusion, Reading Guidelines

In this chapter we have presented the 4WARD System Model, as well as the Architecture Pillars. The Architecture Pillars in turn point the key results of 4WARD, and a brief introduction was given to the concepts and technologies that make up the foundation of those pillars. The 4WARD System Model, through the definition of the Architecture Pillars, defines what can be understood as 'cornerstones' of what will be a more precise definition of an architecture for the Future Internet. Such an architecture will likely include also other building blocks in order to provide a complete and suited architecture for any type of network that would make up a part of the Future Internet.

The different elements and aspects of the 4WARD System Model are further described in Chaps. 4 through 10. Chapter 3 provides a description of the business, socio-economic, and regulatory aspects of future networks which gives important understanding of the interplay between business models, technical development, user needs, as well as regulation and governance. Chapter 11 provides a use case to apply the 4WARD System Model in order to analyze a specific business scenario as to derive a suitable network architecture for that scenario. Finally, Chap. 12 gives an overview of the various prototypes that have been implemented for the purpose of evaluating and demonstrating the 4WARD concepts and technologies.

References

1. M. Achemlal, P. Aranda, A.M. Biraghi, M.A. Callejo, J.M. Cabero, J. Carapinha, F. Cardoso, L.M. Correia, M. Dianati, I. El Khayat, M. Johnsson, Y. Lemieux, M.P. de Leon, J. Salo, G. Schultz, D. Sebastião, M. Soellner, Y. Zaki, L. Zhao, M. Zitterbart, 4WARD Deliverable D-2.1: Technical Requirements (Apr. 2009), http://www.4ward-project.eu
2. OSGi Alliance, http://www.osgi.org

http://www.springer.com/978-90-481-9345-5

Architecture and Design for the Future Internet
4WARD Project
(Eds.)L.M. Correia; H. Abramowicz; M. Johnsson; K. Wünstel
2011, XXIX, 306 p., Hardcover
ISBN: 978-90-481-9345-5