ANA Project

Autonomic Network Architecture

Sixth Framework Programme
Priority FP6-2004-IST-4
Situated and Autonomic Communications (SAC)

Project Number: FP6-IST-27489

Deliverable D.2.2

First Draft of the Functional Composition Framework

Version 1.0

ANA Project

Autonomic Network Architecture

Project Number	FP6-IST-27489
Project Name	ANA - Autonomic Network Architecture
Document Number	FP6-IST-27489/WP2/D2.2
Document Title	First Draft of the Functional Composition Framework
Workpackage	WP2
Editor	Manolis Sifalakis (ULancs)
Authors	Manolis Sifalakis (ULancs) David Hutchison (ULancs) James Sterbenz (ULancs and KU) Tanja Zseby (Fokus) Kave Salamatian (UPMC)
Reviewers	Christophe Jelger (UBasel)
Dissemination level	Public
Contractual delivery date	15th February 2007
Delivery Date	February 2007
Version	1.0

Abstract:

This document considers the design of a functional composition framework with cross-layer control loops that will constitute the basic network subsystem in the ANA node (essentially a replacement for the traditional network stack) by means of which it should be possible dynamically and flexibly integrate new and evolving network functionalities and carry out its autonomic optimisation and reconfiguration tasks. An analysis of its building blocks and their functionality is presented along with an overview of how functional composition will be leveraged by means of the MINMEX architecture in the ANA Node.

Keywords:

Functional Composition, Information Sharing, Cross-Layer Optimisation, Information collection, Measurement Functional Blocks, Composition.

Executive Summary

The goal of the ANA project is to explore novel ways of organising and using networks beyond legacy Internet technology. The ultimate goal is to design and develop a novel network architecture that can demonstrate the feasibility and properties of autonomic networking. As specified in the description of work, it is the intension of the project to address the self-* features of autonomic networks such as auto-configuration, self-organisation, self-optimisation, self-monitoring, self-management, self-repair, and self-protection.

In order to meet the self-* goals set, it is important to enable flexible adaptation of the ANA node operation in response to the application requirements (mission) and administrative policies as well as the environment changes. At the same time collaborative functionality should be easily achievable among the compartment members to provide both redundant and/or collective servicing.

To this end the aim of this deliverable is the analysis, design and prototype development of a cross-layer function composition framework and its comprising building blocks that will leverage and enable the flexibility and functionality precipitated in the ANA architecture. A prototype should abide and comply with the ANA Node Blueprint (D1.4/5/6v1).

Table of Contents

1 Introduction 1
1.1 Scope of Deliverable 2
1.2 Structure of the document 2
2 Terminology 3
2.1 Abbreviations 3
2.2 Definitions 3
3 Functional Composition and Cross-Layering – Requirements 6
3.1 Current Architectural Limitations 6
3.2 Required Functionality for ANA 7
4 ANA Cross-Layered Functional Composition Architecture 10
4.1 Overview 10
4.2 Interfacing with MINMEX 10
4.3 Architectural Building Blocks 12
4.3.1 Compartment Specification 13
4.3.2 The Composition Framework 13
4.3.3 Cross-Layer Functional Composition Model 17
4.3.4 Composition Logic 18
4.3.5 Information Sensing, Sharing and Optimisations 19
4.3.6 Shareable Information: Metrics, Measurement and Monitoring 24
5 Conclusions 32
6 References 33
Appendix A 36

1 INTRODUCTION

The main aim of this task is the design and development of the framework that will constitute the basic network subsystem in the ANA node, by means of which it will be possible to dynamically and flexibly integrate new and evolving network functionality and carry out autonomicity-driven reconfiguration. During the first 18 months of the project this task focuses on the following areas:

- A methodical analysis of the lessons learned from traditional strict-layered architectures in order to identify the cases where layering provides an acceptable solution, and where this model is insufficient. From the results of this analysis, it will be decided which functions, abstractions and other aspects of layered models, will be incorporated in the design of the ANA communications system.
- A study of various cross-layers and functionally composed research proposals from the literature, in order to understand how to best exploit their properties and usage patterns in our design (e.g. dynamic reconfiguration, late binding, function migration, etc). The results of this study will be reflected in the design specification of the ANA composition framework.
- Construction of an ns-2 simulation model of a cross-layered functionally-composed end-to-end protocol, within which various mechanisms and tradeoffs can be explored to evaluate their effectiveness in general, before implementation in the ANA node.
- Prototype implementation of a flexible, reconfigurable, cross-layered protocol suite, which will support the dynamic, ad-hoc and on-demand (re-)composition of the required network functionality. Functional composition will be assisted by means of sensing the environment (instrumentation or *knobs*) that driveg cross-layer optimisations, and based on state information collected from different parts of the subsystem or the network environment, including information from above (*dials).*

Starting from the conceptual work carried out in tasks 1.3, 1.4 and partly 1.5, this task focuses on exploring new flexible ways of componentising the communication functions, and develop a framework based on which these components can be integrated in a meaningful way, so as to dynamically compose a variety of network-based services.

The resource discovery mechanisms that will be developed in task 2.3 will provide ways of discovering new service components in the environment. The protocol lookup operation that will be developed in task 2.4 will permit the creation and maintenance of dynamic paths between two or more components. Resilience features will be incorporated *inline* (as a native functionality), based on principles investigated in task 3.3 and 3.4, while monitoring capabilities and self-management functions will also be supported based on the outcomes from tasks 1.3 and 3.1.

1.1 Scope of Deliverable

The aim of this deliverable is the analysis, design, simulation, and prototype development of a cross-layered functional composition framework that will leverage and enable the flexibility and functionality in the ANA architecture. The prototype will follow the framework of the ANA Node Blueprint (D1.4/5/6v1).

To serve the autonomic purpose advocated in the ANA proposal it is important that functional composition and cross-layer control loops reflects the need for context-awareness, dynamic adaptation, reconfigurability and autonomic optimisation of the systems operation with regard to its mission, application, administrative policy above, as well as to the communication environment an traffic state below.

To this end it is within the scope of this deliverable to explore the need and propose prototype mechanisms by which the ANA node can sense its operational environment, and facilitate optimizations in its basic operation, as well as investigate the type of information needed to drive these optimizations and consider mechanisms for collecting and using the required information.

1.2 Structure of the document

This document is organised in the following sections. Section 1 provides an introduction to the objectives and aims of this deliverable, defines the research scope of task 2.2 and explains how it relates to other deliverables of the ANA project.

Section 2 introduces the terminology used in this deliverable, abbreviations and definitions.

Section 3 is a short requirements analysis of the problem space so as to identify the properties and features required for effective functional composition within the ANA problem space.

Section 4 introduces the proposed architecture for facilitating functional composition, and looks in more detail some of the technical aspects of it. Intentionally, there has been an effort to maintain a relatively high level goal-driven description supported by technical examples (as opposed to following a specification style technical annex) so as to avoid prescribing “finalised” solutions at this stage. A prototype satisfying these objectives and based on these examples is anticipated in the future, that will be proof-tested in a testbed environment and extended before reaching a final specification.

Finally Section 5 concludes this deliverable, by summarizing the information provided in this deliverable.

2 TERMINOLOGY

2.1 Abbreviations

ANA	Autonomic Network Architecture
API	Application Programming Interface
BP	Bootstrap Protocol
FB	Functional Block
IC	Information Channel
IDP	Information Dispatch Point
IDT	Information Dispatch Table
MC	MINMEX Controller
FC	Functional Composition

2.2 Definitions

Identifiers:

- An Identifier consists of a finite sequence of symbols of a given alphabet.
- Identifiers are used for identification of an entity within a set of entities (e.g. to single out one or more objects from a set of objects).
- Identifiers are typically persistent and globally unique within a given identifier space.
- Within ANA, identifiers are used to identify objects, resources, etc.

Names:

- A Name is a globally unique, persistent identifier used for recognition of an entity (e.g. object, resource, and host). For example, a name can be used to resolve the address/locator of the entity.
- Within ANA, names are used to identify an entity (e.g. object, resource, and host). Names can thus be used to resolve the identifier or address/locator of an entity, which is necessary to gain access or communicate with the entity.

Addresses:

- An Address is an identifier that is used for routing and forwarding.

- Addresses entail the information that is needed to transport data/information from a source to a destination.
- In case of structured addresses, which define the location of an entity within a topology, addresses are also called Locators.

Functional Blocks (FBs):

- FBs are the information processing functions in ANA.
- They generate, consume, process, forward information.
- FBs run on one node only, which means a node has full control over the functional blocks it hosts.
- They can have zero or more input and output.

Information Channels (ICs):

- ICs are the channel/medium over which communication between functional blocks takes place.

  Note: ICs can be logical, i.e. they can span across multiple nodes: it is a kind of "distributed object" (made of FBs and other "lower" ICs).

Information Dispatch Points (IDPs):

- IDPs are the entities that provide decoupling for ICs and FBs. That is, access to an FB or IC is done via the IDP it is bound to.
- IDPs allow dynamic (re-)binding of FBs and ICs in a way that is transparent to the "client" of the FB or IC. That is, the entity bound to an IDP can be changed but the IDP used to interface with the entity remains.
- IDPs perform no processing except data forwarding (actual processing of data is performed by FBs).

Compartments:

- Compartments are non empty sets of Functional Blocks (FBs), or non empty sets of composed FBs (where a composed FB is two or more FBs hosted by the same ANA node), agreeing on some common set of operational and policy rules (the "recipe").
- The common recipes are typically the communication principles, protocol(s) and policies to be used.
    - Examples of common communication principles: how naming, addressing, routing, etc. is handled
    - Examples of protocols: communication protocols between peer FBs on different nodes, etc.

    - Examples of common policies: membership (join/leave) procedures trust, etc.

- FBs forming a compartment communicate through Information Channels (ICs). ICs can be seen as abstractions of “connections” or “communication channels” through underlying compartments. In case there are no underlying compartments, ICs represent the underlying communication channels (outside the ANA world) that are used for transferring information between FBs (e.g., an inter-process communication channel, a L2 channel).
- Some compartments may require a minimal set of (composed) FBs in each and every participating ANA node. This is considered again as part of the compartment policies. Those composed FBs are closely related FBs interconnected through IDPs.

3 FUNCTIONAL COMPOSITION AND CROSS-LAYERING – REQUIREMENTS

3.1 Current Architectural Limitations

Layered network architectures have been a powerful abstraction that has allowed technological advances in the Internet, in particular the rapid evolution of link layer an LAN technologies, as well as improvements to TCP without impacting the IP protocol infrastructure. However, the strict layering has also been problematic for a number of domains, including mobile wireless and sensor networks.

This is often reflected in the current applications design trend whereby typically the applications and the user expectations adapt to the network subsystem limitations, a situation that makes e.g. QoS and resilience rather difficult and cumbersome to achieve.

The classical approach to combine functionality in networks has been the use of *protocols*. In general terms, a protocol is nothing more than the formalization of the collaboration between two or more network entities. Classically protocols have been founded on the basis of layers; each protocol defining the interaction between same level layers in the network architecture. Each layer is shielded from upper and lower layers and was providing functionality needed by higher layer protocols only through some predefined interfaces (Service Access Point in the OSI model). This approach has been very successful as it enables independent development of each layer from others. However it has been found to limit performance and flexibility. Opaque layers have appeared as impediment blocking access to internal layer functionality and information where this access could have been useful. The rigidness of the inter-layer interface results also in loss of performance because of the overhead resulting from adapting every transferred frame to be conforming to layer interfaces, and due to the inability to make correct decisions on implicit assumptions about the behaviour of adjacent layers. Recently, an increasing number of proposals have been advocated to relax the layer boundaries and to do cross layer optimisations. Through this, horizontal peer-to-peer relationship between elements sitting at the same layer is extended vertically with cross-layer control loops to a more flexible framework in which components at different layers can collaborate to optimise performance. However, up to now most cross-layering approaches have been based on specific optimisations (such as corruption/loss discrimination in TCP over wireless links e.g. [KS+2004]) without an adequate understanding of the larger architectural impact and feature interactions. Therefore we need to step back and take a broader view of the benefits (in overall performance) vs. costs (in increased complexity and reduced performance due to feature interactions) of cross-layering [S2006].

As the main objective of an autonomic network infrastructure is the localisation and optimisation of the infrastructure to the environment and user needs, there needs to be a

fundamental shift in the design and engineering of the network subsystem, taking into account the dynamic network environment and service requirements. As the rigidity and opacity of traditional layering is challenged by new application scenarios and network scenarios, a number of cross-layer optimisations have been proposed. However, only a few of them can be considered either flexible enough or suitable for facilitating autonomic behaviour in one way or the other.

The ANA infrastructure consists of a set of functional blocks (components) that can be tightly or loosely coupled, either in a single ANA node or distributed over the network. These components reside in a single layer of the traditional architecture or span among several layers. Nonetheless, these functional blocks cooperate to deliver information from one point of the network to another. The functional composition framework in ANA architecture aims toward providing the basic mechanisms for enabling such cooperation between components. This means that cross-layering should not be merely an addendum to compositional framework but an intrinsic property of it. Moreover, the composition framework should be flexible enough to accommodate existing and even future type of cooperation schemes and not act as an impediment for future network developments.

There are three possibilities for the organisation of functional components for communication. The first is the strict layering of the OSI model and Internet protocol suites. The second is the other extreme in which there are no layers, but rather pools of functions that are arbitrarily interconnected in a directed graph. While this approach appears to be attractive due to its flexibility, is hard to manage as the component composite increases. It also limits technological advances as, as the lack of separation of concerns does not permit abstraction of functionality alon well known boundaries, such as the link layer. Finally it is likely to be more difficult to design and engineer resilience and security in large, complex, monolithic systems.. Finally the third approach, which is the one we aim to explore in more depth as part of this task and ANA in general, is one in which strict layering is relaxed without eliminating them all together, resulting in functionally composed protocols with fuzzy layer boundaries across which cross-layer control loops operate. *Dials* expose characteristics below and *knobs* influence behaviour from above..

In this document, we examine the specific requirements that define the design space of a cross-layered functionally-composed protocol subsystem that is able to exploit and support autonomic operation. Based on these requirements, we will design an architecture that combines novel features and frameworks with best practices and techniques from the literature, in order to successfully serve the aims of the ANA.

3.2 Required Functionality for ANA

As the roles of network nodes are expected to vary over time in self-organising network environments, the overall functionality provided by a node needs to be constructed in a modular way (as opposed to a monolithic) and deliver any service as a composite of functional components that are dynamically re-configurable and re-composable at run-time. New functions, protocols, services, and other software extensions should be possible to incrementally and dynamically integrate in this composite.

It is worth noting that the aim is not to completely abolish the useful abstractions provided by layered communication systems, but rather to design a more *modern* network subsystem that while maintaining the benefits of a layered design (by studying the lessons learned and experiences gained through its use in the past two decades). Moreover such a system will extend and enhance its functionality by integrating features and developments from alternative designs such as component and object based paradigms and models.

As dynamic (re-)configurability is the most prominent feature and fundamental requirement in such a modern network subsystem that will encompass autonomic behaviour [SSH06] we need to look into the considerable volume of prior work on component based models such as [Coul03][PP93][JB00] as well as on active and programmable networks such as [SCSS06][MBCSCZ99][WGT98] that have delivered a multitude of paradigms, models and mechanisms in this area.

On the other hand an area where more ground-breaking research and work is needed is in understanding and developing the means that will drive adaptation as well as how to effectively and safely use any such framework without violating the operational correctness, jeopardizing stability, or degrading performance. As a result the scope of this task spawns to include the study of the following aspects as well.

- How to relax the opaqueness of layers or functional entities in a network subsystem by enabling information sharing that can drive optimizations and inter-layer/component control loops. In doing so it is important to warranty that modularity is not compromised and complexity is not increased to the extent that impacts scalability.
- How to monitor and safeguard correct operation (with regard to the goals) and improve the performance of the overall system in face of potential feature interactions among objective-competing optimizations. Most work on cross-layering today has focused either on optimizing operation of a network subsystem based on a single cross-layer optimization or providing cross-layer information sharing frameworks where the optimizing logic resides externally to the system and therefore cast away the issue of caring about correctness. However, for any such solution to become realistically usable, this problem needs to be addressed.
- Ways of determining and enabling vertical (across the stack) composition. It is well known that a full TCP/IP or other protocol suite is not necessarily needed in any environment. For example in a small sensor or ad-hoc network which is seen as a flat topology all link-layer and network layer functions can be collapsed in one layer. If FEC and ARQ are available at the link layer it may be possible to make the transport layer redundant too. Hence, there is a need to engineer ways and mechanisms for assessing the functions needed to deploy in a protocol suite. Moreover being able to do this in a dynamic way (either while bootstrapping or at run-time) makes it useful when the network role of a node changes dynamically.
- Ways of enabling horizontal (micro-protocol) composition. Typically, protocol implementations are based on a set of finite state machines (FSM) that collectively provide a set of semantically-related control plane and data plane functions (end-to-end versus hop-by-hop error control, flow control, congestion control, check-summing, etc). Very often in the name of modularity and isolation many of these micro-protocols are repeated across the stack at more than one

layers (e.g. ARQ in TCP over GPRS) thus introducing unnecessary overheads and yet other times their existence is blindly (falsely) assumed (TCP assumes error free transmission at the Link Layer). To address this problem a less rigid and tightly coupling of the protocol functions can be promoted and advocated in the design of network protocol, which could allow independent/optional use and integration of these micro-protocols as well as the replacement of one implementation with another, subject to requirements. Several efforts towards such direction exist in the literature, for example [DH98], [Feld94] and [Ches98], etc. It is likely that even some of the current protocols could be re-written to favour such a model. Using cross-layer/component information sharing can then allow optimised decision making of what protocol function to enable at which level, so as to eliminate unnecessary redundancy and improve performance.

- Determine data path selection. Once all the protocol functionality for intra-compartment communication is in place, there comes the time of establishing communication with other peer hosts within and across compartments. In a common globally configured environment like today's Internet this has not been a problem because the network subsystem is fixed. However in an autonomic network environment which is expected to promote diversity for the sake of localised optimisation the problem of selecting a path across a multi-protocol capable stack can be a real challenge. A similar (yet not quite the same) problem existed in the past when multiple protocol suites were available for use by different applications. Unlike in the past though where the different protocol suites were competing for their dominance over one specific environment, the future is more diversity-friendly thus promoting the co-existence of heterogeneous protocols in equally diverse network environments each tailored to a specific application domain. Given all this multiplicity and diversity (number of layers, micro-protocol configurations, set of protocol suites, etc) when two peers need to communicate there is need to determine a suitable data path across the network subsystem (partly common with the next hop, partly with the local compartment, and partly with the remote end). A challenging research objective is to develop mechanisms that will allow deterministic selection of suitable data paths across a set of network functions within a node or in a gateway that translates between dissimilar protocols of adjacent compartments.

These research areas impose a set of requirements for the functionality anticipated in ANA. Any dynamically composable or component based platform which satisfies them could then provide the ANA node functionality.

4 ANA CROSS-LAYERED FUNCTIONAL COMPOSITION ARCHITECTURE

4.1 Overview

Figure 1 shows the conceptual placement of the functional composition architecture within the ANA node framework (D1.4/5/6v1).

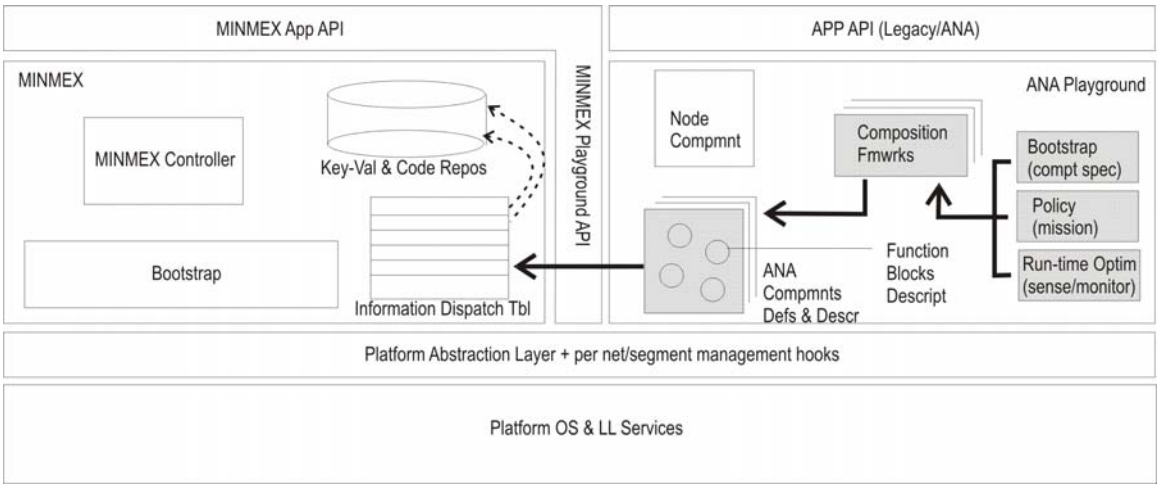

Figure 1. The Functional Composition Framework in the ANA Node

Essentially most of the building blocks of the functional composition architecture reside in the ANA node playground and use the MINMEX services to instrument the assembly of the composed functionality. Practically all the decision making and composition logic as well as its support functions take place in the playground area, up to the point where a function composite is generated and passed to the MINMEX part. There the function composite specification is used to produce the Information Dispatch Table (IDT) with the function IDPs.

4.2 Interfacing with MINMEX

To understand better how the functional composition architecture interfaces with the MINMEX framework we need to first sketch out the functionality of the internal forwarding module in MINMEX. This is illustrated in figure 2.

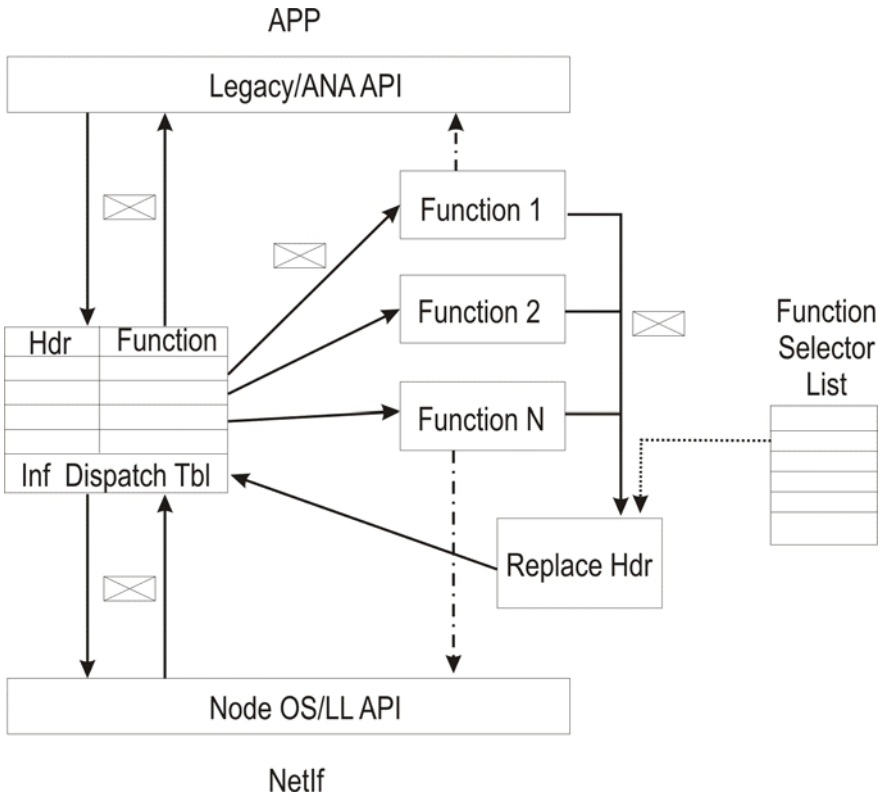

Figure 2. The Internal Forwarding

The interfacing structure between the functional composition framework and the rest of the MINMEX is the function selector list, which is generated by the functional composition framework and essentially determines which functions and in what order will process a packet.

As datagrams flow through the MINMEX framework either from the user application or from the network they are "classified" through the internal forwarding module. Their header (function selector) is looked up in the IDT and then dispatched to the respective function for processing. After the function has processed the packet the current selector is replaced with another one that corresponds to the next function in the processing packet/flow processing chain (by consulting a selector list), and hands it back to the internal forwarding module for the next classification round. When the selector list has been exhausted the packet is routed either "downwards" out of the node to the network using a default forwarding function – by "default" we mean one that is not selected through a selector header (which in turn selects the corresponding outgoing interface), or "upwards" to be dispatched to an application.

The packet does not have to exhaust the list of selector headers before it is handed (down) to the network or (up) to the application. Instead at any moment in time either a copy of the packet or the actual packet may be "routed" out to the network through an interface or dispatched to an application by any of its processing functions, as long as they have the appropriate access rights to the corresponding API, and for the respective operations (*copy-and-send*, or *send*).

In its simplest form this data structure is a simple set or linked-list. However, more sophisticated forms are also possible (such as trees or digraphs) that enable conditional or parallel (pipelined) processing of a packet by a (subset of) function(s).

Access control enforcement to the selector list enables run-time (re-)composition by selectively permitting some functions or other ANA node processes to modify it.

Finally a number of selector lists is possible to exist simultaneously allowing functional composition to be performed at various granularity levels such as per compartment, per (aggregate) flow, or even on session.

An interesting performance experiment for ANA nodes that experience one-time composition, only during the compartment association phase (e.g. high-end routers or single-application small-scale sensors), might be to try and integrate the selector list and the selector replacement operation inside a dynamically generated machine language based, internal forwarding module. After that, getting rid of the functional composition framework or even the whole playground could leave the node with only a minimal code footprint (MINMEX) sustainable on devices of any scale.

4.3 Architectural Building Blocks

Figure 3 shows the main building blocks of the Functional Composition Architecture and their relationships.

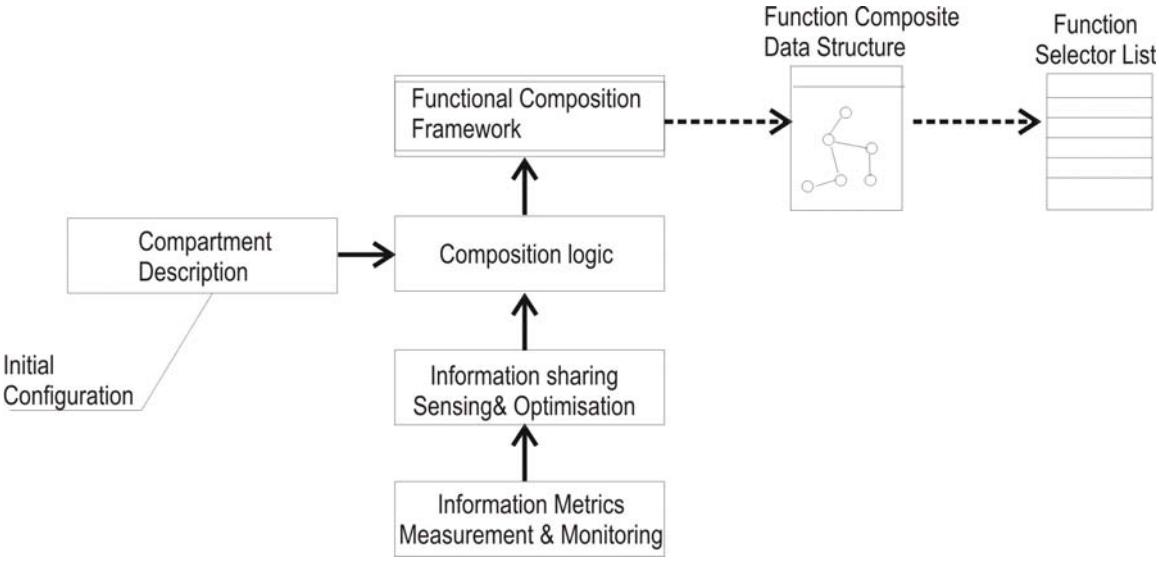

Figure 3. Functional Composition Architecture Building Blocks

The composition framework generates a (set of) IDPs list(s) that are made available to the MINMEX for dispatching packets to the correct functional blocks.

The initial skeleton of functional blocks is determined by consulting a compartment specification, typically during the compartment association phase. At the same time the same specification provides information about possible information sharing interactions and control functions that will drive adaptation. The composition logic as well as protocol specifications, policies and user requirements/constraints (which are provided either at compartment association time or at session initiation) dictate rules and heuristics for the

composition process, the operational constraints and a sound structure for the composite. At run-time automatic adaptation or re-configuration is possible using input from the established information sensing/sharing interactions, emergent optimisations, or changes in the policies/mission. The information needed for the optimisation operations is provided by information flow hooks (D1.4/5/6v1) and monitors (D3.1) available in the system, for collecting the respective data.

4.3.1 Compartment Specification

The compartment specification provides a bootstrap description of the compartment requirements so as to drive the synthesis of the basic compartment functionality. It may include information about functional blocks, layers, cross component interactions, APIs, planes, access control, trust relationships, etc; practically any type of information stemming from the definition of an ANA compartment that can be useful for composing the intra-compartment communication mechanisms and possibly inter-compartment peering relationships.

4.3.2 The Composition Framework

This component is the central one in the architecture as it provides all the capabilities and tools for composing the functionality of the ANA node to operate within a compartment. Practically this framework gathers input from the other components of the architecture and generates the list of functional blocks (i.e. the IDP list) that MINMEX uses in its internal forwarding. The list of functions that are “glued” together by the composition framework include both data plane (packet processing) as well as control plane functions (measurements, monitoring, optimising, etc.).

In practice for ANA to be flexible and evolvable it should be possible that multiple composition frameworks can either co-exist or replace one another in the playground (to satisfy emerging needs or extend the advocated functionality). Therefore no default composition framework needs to be precipitated.

Nonetheless, here we describe a prototype design, of one that exemplifies and demonstrates the expected functionality and fulfils the architectural requirements for ANA with regard to past experiences for extensible functionality in present networks.

Figure 4 outlines the stages and modules where composition may be featured in the ANA operational sphere.

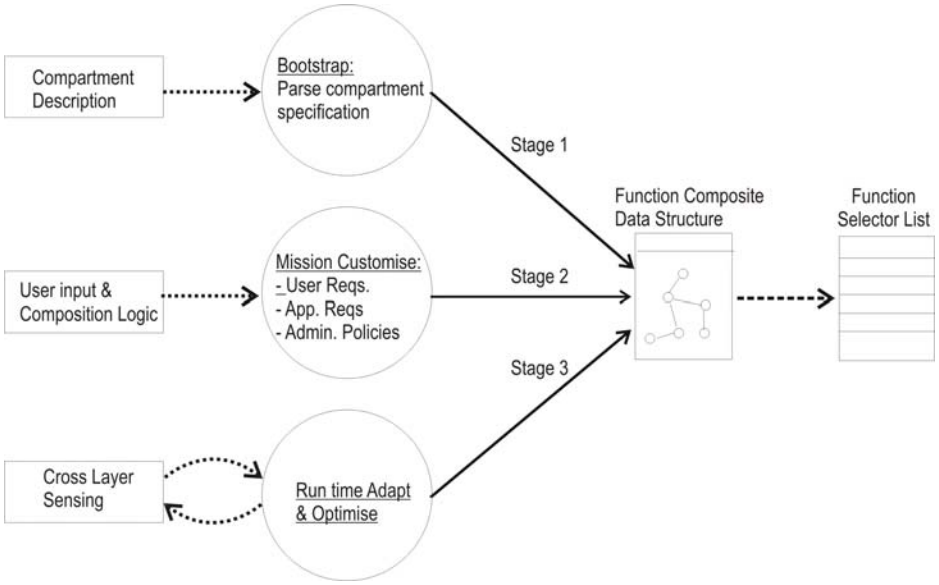

Figure 4. Functional Composition Steps

At each of these steps/stages composition serves different needs and purposes. Initially at the compartment bootstrap process (stage 1) the compartment specification is parsed and based on the information provided there (communication semantics, addressing schemes, routing models, control/data plane multiplexing, layer semantics, APIs, etc) an initial composition takes place providing something like a basic (skeleton) network subsystem. Composition of the sort takes place only once throughout the uptime of an ANA node.

At the second stage, the function composite is enhanced taking into account user input by means of policies and application requirements that essentially prescribe a certain "mission". The composite essentially needs to enrich, constrain or heuristically bound the provided functionality by integrating functional blocks either at the control plane (e.g. to supervise) or data plane (e.g. to optimize) of the ANA node operation. This step may occur several times (not periodically) throughout the uptime of a node as the mission objectives and the application requirements change.

The last and possibly most frequent stage is the one during which re-composition takes place in response to internal state changes, optimisations, or external operational environment changes rendering the system runtime adaptive. This step can be periodic if it needs to take place in response to a periodic event, or ad-hoc if an unexpected or stochastic event triggers it. Due to the frequency that this step might occur, it is important that lightweight and low-latency procedures are adopted during the update of the MINMEX to reflect the changes in the function composite. To this end several strategies may need to be combined to enable controlled and fast updates, such as memory mapping instead of copy operations, fine grained access to the MINMEX functions that handle control plane and data plane functions, or monitoring of data processing functions, and cluster based access to the MINMEX, where parts of the IDT are updated while other parts are read. Overall this aspect is a challenging one to address.

When it comes to the actual composition aspects we differentiate between two main cases, where composition may be required to maximize flexibility and often maintain correctness. We use the terms *micro-composition* and *macro-composition* to distinguish the two and at the same time to capture the fact that one of them takes place at the protocol/function level while the other takes place at the micro-protocol level.

As stated *micro-composition* takes place at the micro-protocol level and it is protocol skeleton driven. The idea here is that a sequence of finite state machines are "glued" together as prescribed in a protocol specification. A protocol skeleton essentially provides the glue for the functions that should be enabled as part of the protocol specification. Different mechanisms implementing a prescribed function or null functions (i.e. functions that perform no particular action) can be inserted or replaced in the skeleton. For example where a protocol specification prescribes the existence of an error correction function it would be possible to insert a FEC implementing function, an ARQ implementing function, or a null function if the function is redundant (e.g. because it takes place at a higher or lower layer) [S2006]. The possibilities for end-to-end vs. hop-by-hop location of such functionalities are shown in Figure 5. Different sets of the possibilities depending on the application requirements (unreliable, quasi-reliable in light grey, or reliable in dark grey that *must* use E2E ARQ) are shown by the circled regions (in a style similar to Karnaugh maps).

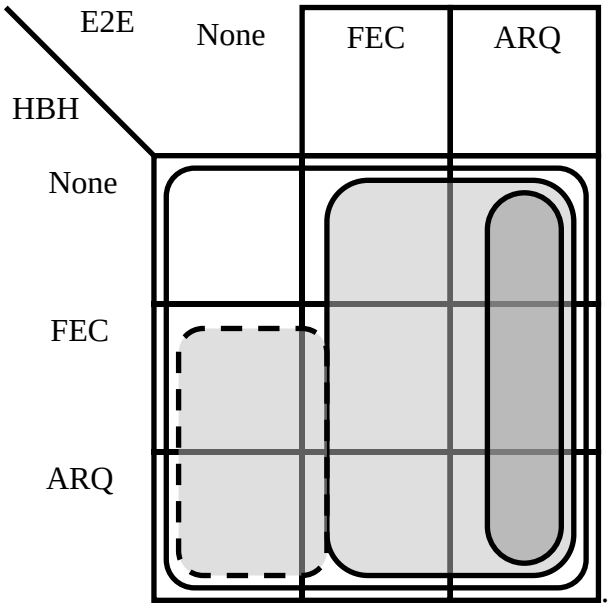

Figure 5. Error Functional Options [S2006]

A few proposals for such composition mechanisms exist in the literature (e.g. [Ches98], [Feld94], [MBCSCZ99]). The main advantage of having such skeletons is protocol correctness, since the strictness of the specification has to be obeyed. For example one cannot insert a congestion control module in the place of an error control module. In representing the implementation of the skeleton as a graph, one can enable the deployment and conditional use of a set of different mechanisms implementing the same function at run time and on a per packet flow basis (a flow may be defined at the application level, transport level or network level). So for example given the three different options available for error correction as depicted Figure 6, one end-to-end

transport level packet flow such as a file transfer may use ARQ, another that is over a high latency satellite link may use FEC, and a lossy media stream may use no-error correction at all. Note that the same graph structure could be representing a link layer protocol skeleton (signifying the functional reuse potential at different levels) whereby the different types of error correction are deployed across different types of physical links attached to the host (e.g. a satellite link, a wireless link and a reliable wired one).

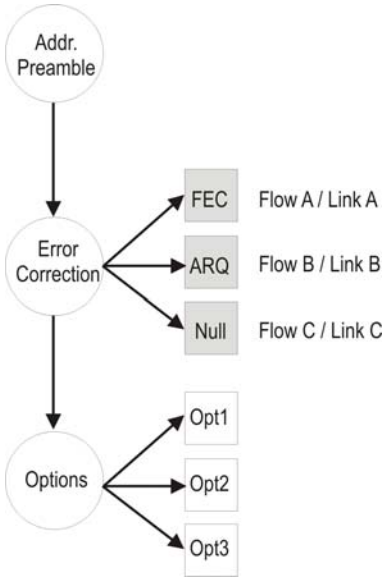

Figure 6. Example of micro-composition

On the other hand *macro-composition* takes place at the protocol or function level. This type of composition allows for more flexibility as there are not strict specifications to be enforced. Examples of such composition are for example layering protocols on top of each other (e.g. layering a network protocol over another network protocol is perfectly fine thing to do), or introducing functionality in between standard protocol behaviour to enhance functionality or include control plane operations (e.g. extension headers in IPv6, measurement/accounting operations, etc). In the literature such flexible composition has been proposed in [SCSS06], [Mor99], [HP91], [CAC93]. This type of composition can also be expressed by means of a graph signifying the path that information flows through as is indicatively illustrated in figure 7 from [SCSS06].

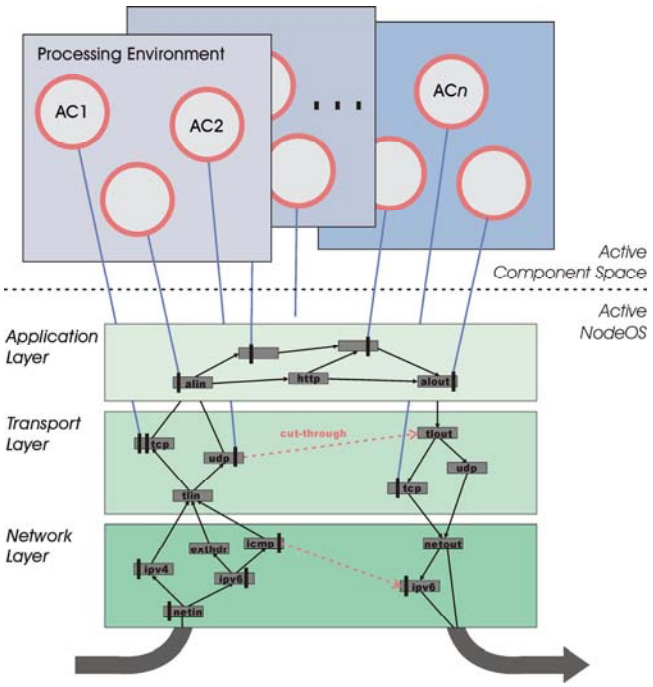

Figure 7. Example of macro-composition from the literature [SCSS06]

Taking into account both types of composition one could represent the whole functional composition process as constructing a two dimensional graph (or a graph of graphs) whereby the first dimension represents the macro-composition and the other the micro-composition.

Finally it is worth pointing out the fact that the functional graph generated from the composition process is not a data path graph for user traffic only! Parts of the graph will typically correspond to the processing path that monitor or control information may follow (e.g. extrapolated network measurements, or monitor data), while other parts will correspond to the actual data path of user flows (reflecting possibly a layered stack). As a result different parts of the composition graph may lead to multiple independent function selector lists that will be facilitated by the MINMEX. The underlying rationale for the separation of the function selector lists reflects the need for associating different access control rights and scheduling priorities to each function selector list.

4.3.3 Cross-Layer Functional Composition Model

An essential step for the investigation of cross-layer control loops and functional composition is to construct a generic simulation model that can be initially used to analyse cost-benefit tradeoffs, and explore alternative approaches to be used in ANA. Ultimately, these simulations will cross-verify results from MINMEX implementation measurements. Figure 8 shows this model.

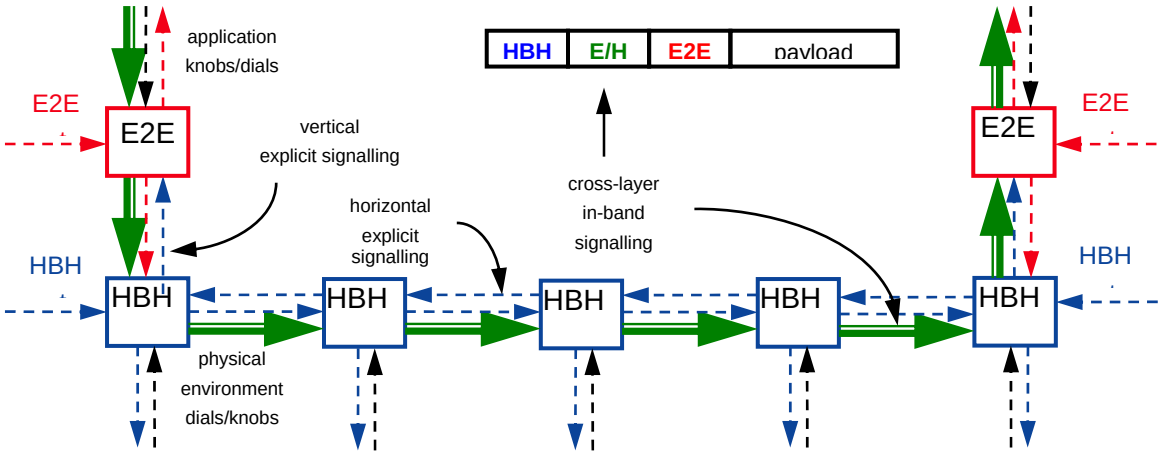

Figure 8. Cross-Layer Function Composition Model [S2006]

The basic model consists of two layers: hop-by-hop (HBH – conventionally the link layer) and end-to-end (conventionally the transport layer). Vertical interlayer functionality consists of multiplexing and protocol encapsulation. Horizontal functionality consists of information flow between sender and receiver (data plane), as well as control plane functionality, in particular transmission rate and error control. Control can be in-band (in the header of data packets) or out-of-band via explicit signalling messages. At each layer, a flexible header contains fields for composed mechanisms. In the case of error control, such as ARQ acknowledgement IDs and FEC redundancy bits. Additionally, cross-layering dictates that some of these functionally composed fields must be visible and modifiable to adjacent layer. In the case of the basic 2-layer model, a field la belled E/H in Figure 8 is the cross E2E/HBH field. The plan is to construct an *ns-2* simulation model of this generic framework as a starting point for the investigation of specific ANA mechanisms.

4.3.4 Composition Logic

This module is responsible for providing the composition logic for generating a function composite. Apart from the algorithmic aspects, it also takes into account the user (end user/administrator) input by means of policies. This logic can be defined at compartment initialisation time or vary in time on a per user session basis.

The main aim of a communication system is to transfer information from one point to destination(s) validating an end-to-end fidelity criterion. A clever resource management of the network should ensure that the fidelity criterion of an application (or a user) is attained with the lowest burden on the network. In very general settings the role of the functional composition framework could be formalized as combining functional blocks that will enable communication and reaching a given fidelity criterion with the lowest possible cost. For example an application would like to attain a given throughput but with the lowest energy consumption. Let's assume that a set of functional blocks $F_1(p_1), F_2(p_2), \ldots, F_n(p_n)$ instantiated with parameters $p_1, p_2, \ldots, p_n$ is used in a communication system. Let's assume also that the fidelity criterion to validate is defined

as $D(F_1, F_2, \ldots, F_n) > D_{\min}$ where the fidelity criterion is indeed a function of functional blocks used as well as parameters used. Similarly the cost function $C(F_1, F_2, \ldots, F_n)$ depends on the functional blocks as well as the parameter used.

The role of the composition framework is to combine components that validate the fidelity constraint and minimize the cost. We will describe this point through an example. Let's assume that in a VoIP application running over a wireless network we need to have a transmission delay less than say 200 msec and we want to reach to this and minimize the cost that is in this case power consumption. The functional composition would have to compose a source encoder, with a routing component and with a MAC/PHY layer. Now let's assume that several source encoders are available each with an encoding delay and a power consumption (because of its processing) and that the power of the MAC/PHY layer could be controlled. The functional composition will have to choose the right source encoder and the right transmission power that will guarantee that the delay constraint is validated and that at the same time the power consumption cannot be lowered. One could imagine that the functional composition will have to trade-off between a higher transmission power at MAC/PHY resulting in less hops and finally a lower delay and more time for the encoder to do the source or a lower transmission power at MAC/PHY layer, resulting in more hops and larger delays, that should be compensated by higher power consumption because of source encoding processing.

The previous example shows the spread of the design space that the functional composer would have to deal with. Constructing autonomic scheme that will enable automatic and adaptive reconfiguration of the functional composition is one of the main challenges of autonomic networks.

4.3.5 Information Sensing, Sharing and Optimizations

The objective of an information sensing and sharing architecture is to drive adaptation and optimisation of what lies between the user (application) and the network environment (i.e. the network-subsystem, be that a layered one, a monolithic one, or component-based one), so as to improve performance and stability. In that respect the (cross-layer/component) sensing and optimisation architecture supports and provides feedback for run-time functional composition and (re-)configuration.

Figure 9 shows a conceptual diagram of the functions, that such a framework is required to be capable of performing.

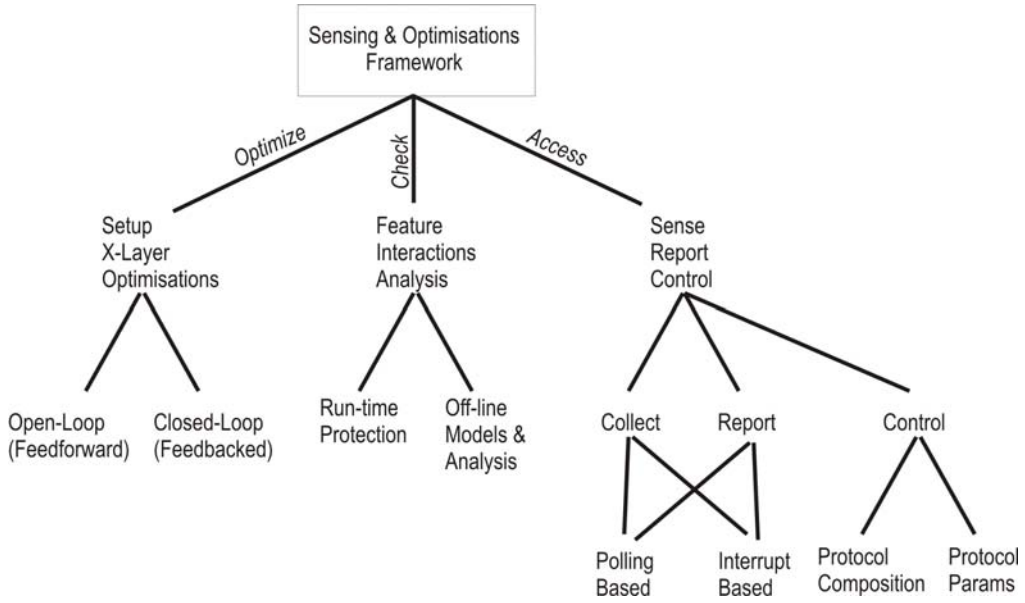

Figure 9 Conceptual Framework

From a functional perspective an information sensing/sharing framework should encompass 3 main capabilities denoted in the diagram by the three labelled main branches as "optimise", "check" and "access".

The "*optimise*" branch essentially refers to the enabling capability for optimisations driven by the sensed/shared information. If such optimisations are modelled as typical control systems with input, processing/control functions and output then lending from control systems theory we are able to classify them in two distinct types, namely *open-looped* and *close-looped*. In open-loop optimisations the processing of the sensed input generates (or triggers) the optimised or optimising output action. Most cross-layer optimisations proposed in the literature so far are of this type flowing either upwards (sensing/input from a lower layer leads to optimisations in an upper layer – e.g. [HTLLS06]) or downwards (requirements input from an upper layer – application typically – lead to adaptation at a lower layer – e.g. [WCZS92]).

In feedback looped optimisations however, the optimised/ing output is fed back in conjunction with the input to further adapt the system and further adjust the output until stability is achieved by evaluating an error function. From a temporal aspect one can simply see the reinforcement of the output back to the input as a way of adding memory of past conditions to the optimised system while from a spatial perspective one can simply think of it as a self-correcting function. Figure 10 illustrates the concept [ST01] for the two cases where by an open-looped optimisation has the form of a controlling knob between two layers, while the close-looped optimisation resembles the combination of a knob with a dial.

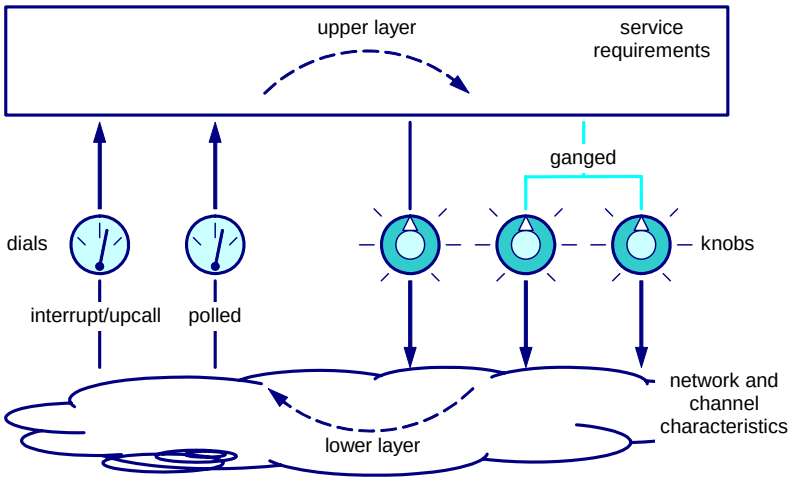

Figure 10. Cross-Layer Knobs and Dials [ST01].

As part of the architecture, the (cross-layer/component) sensing and optimisations framework will provide an API through which an optimisation function will be able to register, set-up input/output hooks, and provide access credentials to the requested layer and measurement metrics. Typically such an optimisation function might be either internal to a micro-protocol finite state machine (e.g. as in TCP congestion control) and therefore in-band in the data path, or external to the data path (e.g. an external control/management component performing out-of-band configuration tasks).

The "*check*" branch involves mechanisms and tools that can be used to verify and check the correctness and conflict free feature interactions in the presence of multiple optimizations. This is a particularly challenging issue and work area where there is not much research in the literature yet (apart from pointing out the problem [KK05]).

There are two types of checks taken into account in the framework as depicted in figure 7. The *on-line checks* involve the employment of a plugin based interface through which "system health" protecting modules can be attached to the framework to safeguard for the system consistency in face of competing interactions either at instantiation time or at run-time (however similar interactions accounted in the compartment specification, off-line checks should be safely assumed). A functionality of one such prototype plug-in is described briefly in the next subsection.

The *off-line checks* on the other hand involve the development of models, simulations and tools for checking and verifying correctness in face of feature interactions off-line before instantiation time. Although this aspect is considered integral in the overall framework, however no actual run-time module is assumed.

The third and last branch labelled "*access*" in figure 7 essentially refers to the capability of collecting information (sensing), reporting and controlling the various configurable aspects of the composition process. *Collecting* and *reporting* of information is assumed to be possible in two possible ways. One is polling-based whereby a shared memory space is used for storing and retrieving data (analogy of environment variables) asynchronously. The second mechanism is *interrupt/notification-based*. A more synchronous operation is assumed in this model and the occurrence of an "interesting" event will trigger the activation of a set of registered call-back functions that will multicast the "news" to the interested parties. A framework providing this functionality is

presented in subsection 4.3.6.2. The *control* capability refers to the access through the framework to the protocol composition (functional composition) and the protocol parameters (protocol tuning), which is instrumented by means of respective configuration APIs. A composable micro-protocol that has tuneable parameters is practically responsible for implementing such an API to enable the cross-layer module to access them. A mechanism enabling this functionality is presented in subsection 4.3.6.4.

4.3.5.1 Protection against Feature interactions

Modelling optimisations as feed-forward/back control systems enables us to use control systems theory and tools to study the issue of stability. In this section we consider a potential mechanism for safeguarding the healthy operation of an optimisation or protocol in a system independently of others in face of potential interactions with them, which is based on the modelling of a control system in the state space.

In control theory state space analysis is based on describing a system's behaviour with a set of so called state variables. The state variables typically capture the input, output and internal state of a system at any moment in time. A set of equations, called state equations depict the dependencies between the state variables. Using the state equations and any current state of the system one is able to calculate what will be the next state of the systems given some discrete impulse signal. For computational simplicity the state variables and equations of a system can be represented using the so called state matrix.

Initially when modelling a control system (i.e. optimisation in our case) the challenge is the selection of a good set of state variables that can sufficiently describe the system. Also this is quite crucial for the successful use of any of the stability test methods as well as for reducing the computational complexity of the state equations.

Once a set of suitable state variables has been determined, the stability of the system can be tested off-line either using simulations or using mathematical methods such as the Routh-Hurwitz criterion [Khat81], the Nyquist criterion [SS65], and other. These off-line tests will also provide a range of acceptable values for each of the state variables, for ensuring the systems stable operation.

At deployment time of the optimisation, the set of state variables can provide heuristics for enforcing boundary conditions in the operation of the optimisation and warranty it is performing within the specification. The state variables would typically represent measurable/monitored of calculated metrics in the sensing and optimisations framework (section 4.3.6.2). When an out-of-range value is detected for a state variable, an event can trigger some response action, which can be as simple as disabling the optimisation, enabling some quarantine condition, or as complex as taking some "healing" measures.

The state variables, their allowed value ranges and the low level information on which they depend as well as the response actions in case of instability may be described as part of the optimisation specification.

Under the same thinking another possible approach would be to consider the whole network subsystem as a complex control system that needs to operate stably at all times. However the complexity of tracking all the interactions and determining suitable state variables, and deriving the state equations might prove to be particularly challenging.

To briefly exemplify the proposed mechanism we consider a case study from the literature [KK05] that illustrates a possible feature interaction. In this example a wireless ad-hoc network is assumed. The optimisation aims to provide optimal performance for TCP by adjusting the number of immediate neighbours seen, through tweaking the transmit power. This is done in two stages with two nested adaptation loops. In the inner loop, a node controls its transmit power so that the number of one-hop neighbours (*out-degree*) is driven to a parameter called *target-degree*. Transmit power is increased by one level if the number of one-hop neighbors is less than target-degree and decreased if it is greater than target-degree. An outer loop sets the value of the target-degree, based on the average end-to-end network throughput. The same action of the previous iteration is repeated if the network throughput increased from the previous iterate or it is reversed if the network throughput is decreased from the previous time step. If the network throughput is zero, the network is assumed to be disconnected and the target-degree is increased again. The outer loop is operated at a slower timescale than the inner loop to avoid the instability and incoherence that results from two simultaneous adaptation loops interacting with the same phenomenon, as elaborated on earlier. The outer loop therefore attempts to drive target-degree to a value that maximizes the end-to-end network throughput, and would therefore eventually drive the network out of any possible state of disconnection. To avoid getting in a complex analysis the following table simply outlines a set of possible state variables for this optimization, the source of information for its estimation and a description of the boundary conditions they enforce.

Table 1. Possible State Variables for Optimisation Sanity Checks

State Variable	Information Source	Remarks
V1 (E2E Throughput)	TCP	Objective is that V1>0 at all times and max while the battery level drop rate does not exceed a max value
V2 (Target Degree)	Optimisation Config	Should not fall below a watermark
V3 (Out Degree)	Optimisation Config	Should not exceed of fall below watermark values
V4 (Power level)	H/W Batt status	Should not reduce at a higher rate than Max-Rate
V5 (Xmitt Power)	PHY	Should not increase above a Max value
V6 (Target Degree Updt Freq)	Optimisation Config	Should not flunctuate above a certain Max-Freq because it will lead to instability
V7 (Out Degree Updt Freq)	Optimisation Config	Used in calculations only

Note that the mechanism proposed in this section, although sound theoretically, remains to be tested from a feasibility and complexity aspect. Moreover it suggests only a sample case of a possible solution for protecting against feature interactions. The information sensing, sharing and optimizations framework is expected to be able to accommodate other candidate mechanisms such as those advocated in the previous section.

4.3.6 Shareable Information: Metrics, Measurement and Monitoring

This module is responsible for establishing the monitoring hooks and measurement metrics that will allow the cross-layer component to collect and share information and use it to enable optimisations and trigger run time adaptation and system health protection.

Work in this area focuses on two fundamental problems. The first one is how to define and specify generic metrics for accessing and acquiring information from the different abstraction levels and how these metrics can map to actual measurable entities in the network subsystem. The second one is how to seamlessly access and acquire the actual information through monitoring and measurement operations.

4.3.6.1 Shareable Information Analysis

Regarding the first of the two aforementioned work areas we did an initial analysis and classification of different measurable metrics that we derived from the existing literature on cross-layer mechanisms, frameworks and optimisations. The classification and categorisation was done based on various aspects, such as the information source (which layer), the type/form of the metric representation (scalar, boolean, event, etc), the access type (read-only, or read-write as are protocol parameters), the dependence to a certain layer or micro-protocol, the scope (layer local versus across layers, and node local versus network wide), validity time, accuracy, and extrapolation approach (accounting, configuration state or inference assisted estimation). The complete list of the metrics considered and their classification is provided in appendix A. Following is a brief summary of some conclusions that emerged from this analysis:

- First of all the vast majority of the metrics as abstractions are micro-protocol specific and therefore reusable across layers as long as the respective finite state machine skeleton is present within a protocol layer.
- On the other hand the source of information of these metrics may differ depending on who is the interested party of that information. As a result for example congestion at the link layer might be detected by seeing lots of collisions while congestion at the transport layer might be detected by looking at the combination of the timeout timer triggers (TCP timeout) and the (lack of) frame errors at the MAC level.
- There are a number of lower level metrics that can be grouped and “masked” under same higher level abstractions, thus allowing a high-level generic abstraction to multiplex a combination of lower level metrics (as already exemplified in the aforementioned bullet point).
- Most of the metric information can be acquired using polling-based access or event-based hooks. However, in the case of event notifications a context needs usually to be provided as well with more detailed information.
- The format/type of the metrics fall in a small range of types that can be defined and provided as an attribute to assist in the retrieval and handling of the context information.

Based on these findings we extrapolated an initial set of requirements for the design of a flexible framework that will allow generic unified and flexible access to information from various parts of the ANA network subsystem, and enable monitoring and optimisations:

- The framework should be flexible to facilitate polling-based as well as interrupt/notification-based access to monitor and configuration data.
- It should be possible for any entity to express interest in a set or combination of measurable low level metrics by means of an abstraction, and receive notifications and context information whenever the information is available
- It should be possible that a set of low level metrics are combined logically or otherwise under a higher level abstract metric and trigger a notification (to the interested parties) when the combined expression is evaluated to TRUE.
- Multiplexing of various combinations of low-level metrics under a higher level metric that is evaluated differently at different contexts (e.g. transport layer versus link layer) is a desired feature.

4.3.6.2 A Framework for Information Sharing

Starting from the analysis in the previous section and taking the requirements into account, in this section sketches out a framework for decoupling meta-information collection from its use, by cross-layer optimisation modules or other entities in the ANA space. This framework rests upon an event mechanism that uses high level metric abstractions to logically multiplex low level (raw) metrics and signal notifications to interested parties. Care is taken to enable information sharing among entities in an extensible way, and yet without restricting flexibility (that would be available if source and recipient of the information where directly communicating).

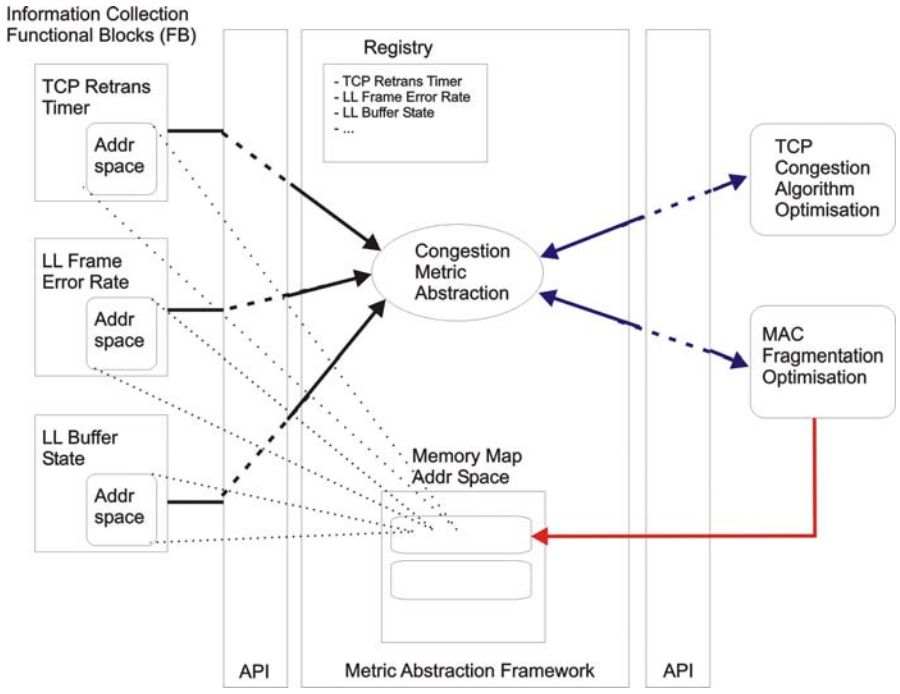

Figure 11. Information sharing framework

Figure 11 illustrates the building blocks of this framework and exemplifies its functionality based on one such abstraction.

Looking at the figure from left to right one can see on the left the functional blocks that are responsible for providing the actual information represented by the low level (raw) metrics. Each such functional block makes known its existence through registration to a metrics registry as seen in the figure. At the right end (service/client side) of the figure one can see the entities which are interested in accessing the information provided by the raw metrics functional blocks. These entities may be either cross-layer aware protocol engines, mediator modules that use the cross-layer information to tweak legacy protocol parameters and data structures, or other client modules (functional blocks) in the ANA space that are interested in the information. Inside the framework resides the raw metrics registry, the instantiations of the high level metric abstractions and the metric state/context memory, where detailed context information about the measured entities is made available.

The operation of the framework can be summarised as follows. All the functional blocks that are available for extracting metric information are registered with the framework. When a client entity comes that needs access to shared metric information it should be able to either provide a specification of the information it is interested in, or provide a higher level metric abstraction name. Given a specification of the high level metric abstraction, the framework instantiates the abstraction which essentially looks up in the registry for the associated raw metrics, “connects” to them by memory mapping their context address space to the metric context memory and optionally combines them logically using an event evaluation function that can trigger notifications to the client (if the client wants to have interrupt-based access to the information).

Note that in that respect the same high level metric abstraction can be used to service different clients by multiplexing different combinations of the same and new low level

metrics. This provides thread/process economy through re-use. For example in figure 9 the metric abstraction named *congestion* is used to service a link-layer module and at the same time a transport (TCP) layer module by multiplexing information collected by 3 raw metric functional blocks (TCP timeout, link-layer error rate, and MAC-level queue state). A client module is notified of the memory mapped address space where the metric related context information is made available to it. If an event trigger function is generated for the serviced entity then a notification is sent whenever the event function evaluates to TRUE, through a call-back mechanism by invoking a registered call-back function. Alternatively, the client can poll in ad-hoc manner the metric context memory for changes and updates regarding the metric data.

Figure 12 shows the underlying "wiring" of event triggers for the scenario of figure 9. The figure illustrates 5 different notifications encoded in the *congestion* abstraction multiplexing the three raw metric functional blocks. They can trigger different optimisations at the MAC layer or the TCP layer according to table 2. The notifications are sent to the two client modules in figure 9. The proposed optimisations are taken from the literature and are described in [KN04] [CSN01].

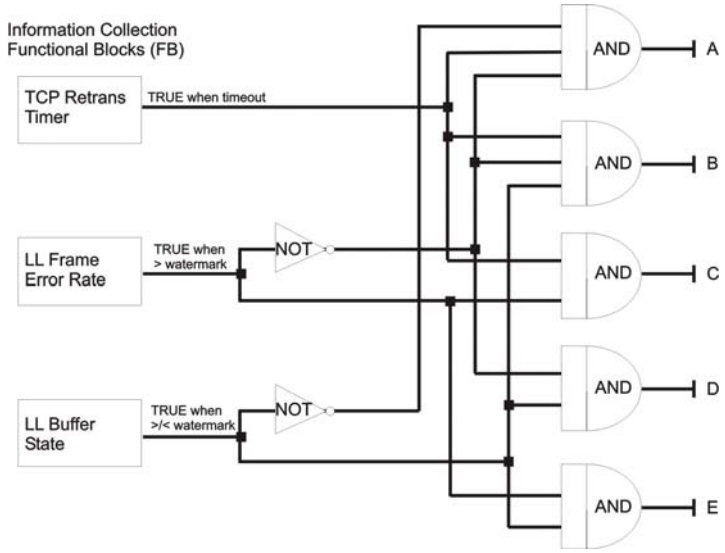

Figure 12. Raw Metrics Multiplexing

Table 2. Notifications and optimisations

TCP Retrans Timer	LL Frame Error Rate	LL Buffer State	Signal	Interpretation – Action
TRUE	FALSE	FALSE	A	Congestion in the network (e2e path). Enable TCP congestion algorithm and possibly do resource reservation.
TRUE	FALSE	TRUE	B	Congestion at the immediate next hop. Enable congestion relaxation at the MAC layer.
TRUE	TRUE	X	C	Errors at the LL. Freeze congestion control at the TCP level. Change fragmentation scheme at LL.
X	FALSE	TRUE	D	Congestion at the LL. Enable channel reallocation.
X	TRUE	TRUE	E	Errors due to interference at the LL. Change MAC fragmentation scheme.

4.3.6.3 Low level information collection

The second work area involves a study of the different possibilities and potentials for performing measurements and monitoring operations in a generic way exploiting the functional composition framework. Essentially, this work focuses on functional blocks that seamlessly (or not) are employed in ANA for collecting and making available for sharing the raw metric information.

Before sketching out possible solutions and mechanisms, we need to break down and analyze the problem at hand in order to acquire more in depth understanding of the requirements. As our main objective is towards engineering mechanisms for collecting information and performing measurements, rather than proposing new measurement techniques, we need to consider what techniques are already available and identify their deployment requirements/limitations.

From the classification in section 4.3.6.1 we have so far identified three different ways of producing the shareable information required for the raw metrics, i.e. by inference/estimation, through accounting, and by accessing fixed state (configuration or other stored state, e.g. connection state).

The latter type of information is probably the easiest one to collect as it is usually maintained in database storage, configuration files, or persistent memory that can be easily accessed locally or remotely (often through some existing API available).

The former two on the other hand (inferred/estimated, accounting) are more difficult to acquire as there is often need for establishing some sort of measurement or monitoring (software or hardware) infrastructure either in-band or out-of-band in the data path. To this end it largely simplifies the problem analysis (permitting a unified model), if we start thinking in terms of functional blocks and perceive the problem as one where perform measurements are carried out in a network of interconnected functional blocks. In this model a physical network is simply one instance and a composite of interconnected software modules (comprising a network subsystem) is simply another). In that respect the problem can be reduced to either acquiring running or configuration state resident within the functional blocks (through some common monitoring API) or carrying out measurements on the communication between functional blocks.

From the available literature we can identify three main approaches of carrying out measurements in a network: (a) *active measurements* whereby one needs to introduce probing traffic for measuring certain characteristics [Dow99][Geo01]; (b) *passive measurements*, whereby one needs to set up monitoring functions at different points of interest in the network and seamlessly extract the useful information from the traffic passing through, then cross-correlate the measurements from the various points and estimate the metrics if interest [ACCTW97][Netf]; and finally (c) *in-line measurements*, by means of which, one can perform semi-active measurements at different aggregation levels and in a non-intrusive way. In-line measurements are based on the principle of either piggybacking or inserting measurement-carrying fields in-lined in packet data, which get processed at various points in-side the network [PHGGS04].

These three different approaches of carrying out network measurements enforce to some extent the bounding conditions of our design space. In general it should be possible to introduce functional blocks (often seamlessly by means of the information dispatch points - IDPs) in an ANA compartment that can perform the following functions

1. Insert, remove and process measurement data in-lined in data traffic.
2. Generate probing traffic addressed to other functional blocks.
3. Collect out-of-band, aggregate and process measurement data from various points in the network.
4. Access other functional blocks and monitor their running state or access their configuration state.

Looking at bullet points 1 and 3 it becomes apparent that quite often in order to provide or generate the raw metric information required, a composite of cooperating measurement functional blocks (as opposed to a single one) will need to be deployed. This is particularly true for cases where the metric data is generated as a result of cross-correlating (and processing) data collected from multiple network points, or cases where differentiating between two or more network points is required to extract the metric information. This imposes the additional requirement that

5. Composition, cooperation and potentially synchronisation (out-of-band signalling) among measurement functional blocks across the network is a desired feature.

Having prescribed the required functionality the question is how this functionality is facilitated by means of the ANA node framework. First of all, expressing the required functionality in terms of functional blocks enables easy generic and lightweight implementation on top of the MINMEX. The need for composition at the node level can be facilitated by the herein proposed functional composition framework, while the cooperation and synchronisation aspects we anticipate are typically tailored to case-specific needs and therefore we expect that the “measurement composite” should be able to prescribe and deploy.

4.3.6.4 Information Hooks in the Processing Function Chain

As pointed out throughout this manuscript thus far, the dynamic adaptation of functionality and parameter optimisation for the deployment environment and mission objectives requires a fast and flexible observation and evaluation of the current internal state and environment conditions. The quality of decisions about functions and parameter adjustments highly depends on precision and timeliness of the observations about the current state of network components. Measurement functions are needed for supporting the establishment of an optimal function chain and later for the adaptation of parameters or functions itself to optimize the operation of the function chain.

We distinguish between two different types of monitoring functions

- Auditing of execution and configuration state: Such functions supervise the operation of a functional block. It is important to control the operation of a functional block, provide statistics about operational variables (e.g. queue length) or current configuration settings (e.g. flow definition) and allows detecting suspicious or malfunctioning behaviour for security reasons (e.g. anomalies in

routing table). Such auditing functions should be attachable to a functional block and be enabled on demand. Those functions can also support feature the analysis of interactions. The integration of such functions in a functional block may be achieved in various ways: as library routines through some plugin mechanism at the playground, some well known API available on the functional block, or through the MINMEX.

- *Traffic and network state measurements:* These are measurement and monitoring functions that observe events happening (e.g. certain traffic patterns) in the network and extract useful information (e.g. QoS parameters) and the state of (parts of) the network (e.g. available bandwidth and topology). Such functions can be realized as independent functional blocks that provide active or passive measurement functionality. Those blocks can be integrated in the function chain at relevant observation points seamlessly by means of information dispatch points (IDPs – see ANA blueprint) and through the MINMEX.
-

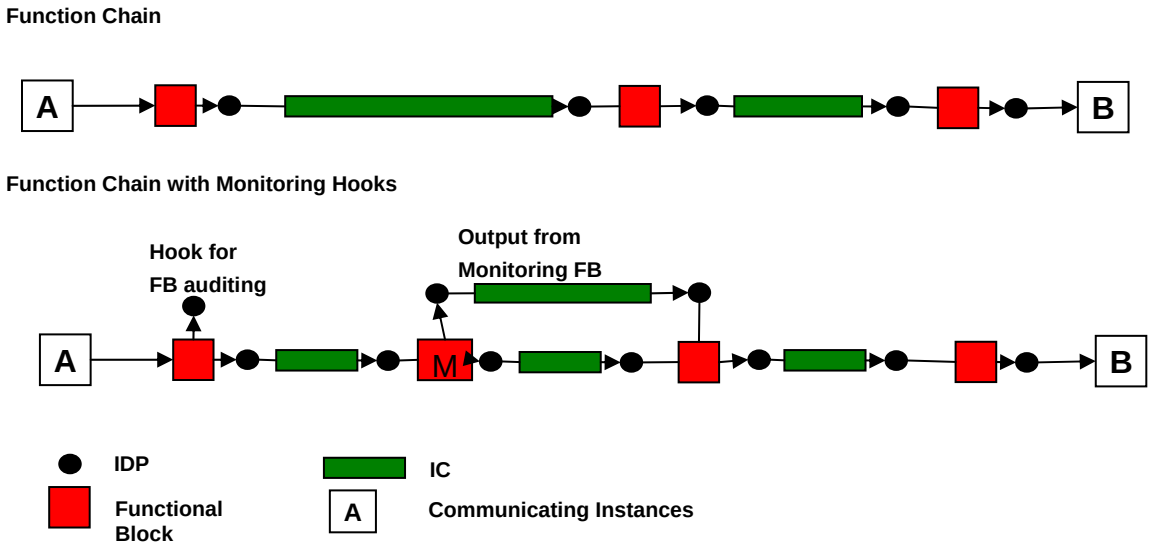

Figure 6: Measurement Hooks in Function Chain

Figure 13 illustrates the different types of measurement hooks in the function chain. Functions for auditing the operations of a functional block maybe be integrated in the functional block itself or provided as an extension to it. The enabling/disabling of the auditing functions is done together with setting the parameters for the functional block. It would be useful if a default setting for auditing is provided that is enabled if no specific parameters are given. Traffic, network, or device-state information can be monitored by separate special measurement functions. Those are specific functional blocks that can be included in the function chain where needed like any other functional block. It is possible to put multiple measurement functional blocks directly after each other in the function chain.

Measurement results from separate functions which may help to set and adjust micro protocol operations and settings (e.g. the measured loss rate triggers the amount of FEC data) or provide input for any cross-layer and intra-layer optimisations and auditing to

adapt the function chain to changing conditions (e.g. introduce another error correction method and exchanging the functional blocks).

Processing of measurement data into derived metrics can be done either in the measurement functions itself or externally in additional functional blocks. If the processing functions are available at the measurement function itself, the derived metrics are offered by this measurement functional block and are part of its portfolio (capability set). If the derived metrics are calculated by other extra functional blocks those blocks may have to be included in-band or out-of-band into the function chain. Derived metrics may need input from multiple measurement functional blocks as explained in the previous paragraph, thus forming measurement function composites (see Figure)

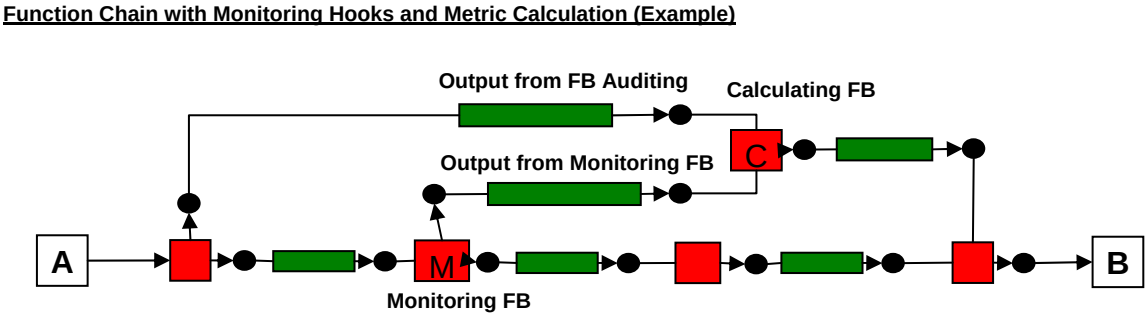

Figure 7: Calculation of Derived Metrics in Function Chain

5 CONCLUSIONS

This deliverable presents the design considerations and the draft architecture for the basic network subsystem in the ANA node (essentially a replacement for the traditional network stack) by means of which it will be possible to dynamically and flexibly integrate new and evolving network functionalities and carry out its autonomic reconfiguration procedures.

Given the objectives and aims of ANA we have tried to identify a set of requirements for a functional composition architecture by means of which an ANA node can leverage the autonomic goal and enable all the necessary functionality.

Work on this deliverable has focused on the aspects of composition, componentisation of functionality, dynamic adaptation driven by sensing internal state and the external environment, and collecting abstracting and using monitor data meaningfully for dynamically optimising the ANA node functionality.

Although in several aspects the proposed architecture lends itself ideas, paradigms and possible solutions from literature, in many other aspects novel ideas and viable solutions have been proposed in this deliverable, which will be tested and deployed throughout the duration of this project. The challenge we are facing in this task and within the ANA project in general is to bring these technologies and mechanisms together, in a functional architecture that will fulfil the autonomic objective anticipated.

6 REFERENCES

[ACCTW97] Apsidorf, J., Claffy, K., C., Thompson, K., Wilder, R., OC3MON: Flexible, affordable, high performance statistics collection, in Proc. of the seventh annual conference of the Internet society (INET'97), Kuala Lumpur, Malaysia, 1997

[CAC93] Russell J. Clark, Mostafa H. Ammar and Kenneth L. Calvert "Multiprotocol Architectures as a Paradigm for Achieving Interoperability", in Proceedings of IEEE INFOCOM '93, San Francisco, April, 1993

[Ches98] G Chesson, "XTP/PE Design Considerations", Proceedings of 13th Conference on Local Computer Networks, October 1988.

[Coul03] G. Coulson et al, "NETKIT: A Software Component-Based Approach to Programmable Networking", in Computer Communication Review, a publication of ACM SIGCOMM, Vol 33, Oct 2003.

[CSN01] Ci, S., Sharif, H., and Noubir, G. 2001. Improving performance of MAC layer by using congestion control/avoidance methods in wireless network. In Proceedings of the 2001 ACM Symposium on Applied Computing (Las Vegas, Nevada, United States). SAC '01. ACM Press, New York, NY, 420-424. DOI= http://doi.acm.org/10.1145/372202.372390

[DH98] Deering, S., Hinden, R., Internet Protocol version 6 (IPv6) specification, IETF, IPNG Working Group, RFC 2460, December 1998

[Dow99] Downey, A., B., Using pathchar to estimate Internet link characteristics, in Proc. of ACM SIGCOMM'99, Cambridge, MA, pp. 241-250, September 1999

[Feld94] D. Feldmeier, "An Overview of the TP++ Transport Protocol Project, in High Performance Networks", Ahmed N. Tantawy editor, Kluwer Academic Publishers, Norwell MA US, 1994.

[Geo01] Georgatos, F., Gruber, F., Karrenberg, D., Santcroos, M., Susanj, A., Uijterwaal, H., Wilhelm, R., Providing active measurements as a regular service for ISP's, in Proc. of Passive and Active Measurement Workshop (PAM2001), Amsterdam, NL, 2001

[HP91] N. C. Hutchinson and L. L. Peterson, "The x-kernel: An Architecture for Implementing Network Protocols," IEEETransactions on Software Engineering, vol. 17, pp. 64--76, January 1991.

[HTLLS06] I. Haratcherev, J. Taal, K. Langendoen, R. Lagendijk, H. Sips, "Optimised Video Streaming over 802.11 by Cross-layer signaling", IEEE Communications Magazine, Jan 2006.

[JB00] M. Jung and E. Biersack, "A Component-Based Architecture for Software Communication Systems", Proceedings of IEEE ECBS, Edinburgh, Scotland, April 2000.

[Khat81] K.J. Khatwani, "On Routh-Hurwitz Criterion", IEEE Transactions on Automatic Control, VOl AC-26, p 584, Apr 1981.

[KK05] Vikas Kawadia, PR Kumar, “A Cautionary Perspective on Cross Layer Design”. Published in IEEE Wireless Communication Magazine on Feb 2005

[KN04] JaeWon Kang Nath, B. “Resource-controlled MAC-layer congestion control scheme in cellular packet network”. In proceedings of 59th IEEE Conference on Vehicular Technology, VTCC 2004, Spring. 2004

[KS+2004] Rajesh Krishnan, James P.G. Sterbenz, Wesley M. Eddy, Craig Partridge, and Mark Allman, “Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks”, *Computer Networks*, vol.46, #3, October 2004, pp. 343–362

[MBCSCZ99] S.Merugu, S.Bhattacharjee, Y.Chae, M.Sanders, K.Calvert and E.Zegura, "Bowman and CANEs: Implementation of an Active Network", Invited paper at the 37 th annual Allerton Conference on Communication, Control and Computing, Monticello, Illinois, September 1999

[Mor99] R. Morris et al, “The Click Modular Router”, In 17th Symp. on Operating Systems Principles (SOSP'99),Ki- awah Island, SC, 1999. ACM.

[Netf] Cisco IOS NetFlow, http://www.cisco.com/warp/public/732/Tech/nmp/index.shtml

[PHGGS04] Pezaros, D., P., Hutchison, D., Garcia, F., J., Gardner, R., D., Sventek, J., S., In-line Service Measurements: An IPv6-based Framework for Traffic Evaluation and Network Operations, in Proceedings of the 2004 IEEE/IFIP Network Operations and Management Symposium (NOMS'04), Seoul, Korea, April 19-23, 2004

[PP93] Plagemann T., Plattner B., “Modules as Building Blocks for Protocol Configuration”, Proceedings International Conference on Network Protocols, ICNP'93, San Francisco, CA, Oct. 19--22, 1993, pp. 106--115.

[S2006] James P.G. Sterbenz, *Towards a Framework for Cross-Layer Optimisation in Support of Survivable and Resilient Autonomic Networking*, Dagstuhl Perspectives Workshop on Autonomic Networking, Wadern Germany, January 2006

[SCSS06] S. Schmid, T. Chart, M. Sifalakis, A.C. Scott. "A Highly Flexible Service Composition Framework for Real-life Networks". Journal of Computer Networks, Special Issue on Active Networks, Elsevier, Vol. 50/Issue 14, p. 2488-2505, 5 October 2006.

[SS65] M.R. Stojic, D.D. Siljak, “Generalization of Hurwitz, Nyquist and Mikhailov Stability Criteria”, IEEE Transactions on Automatic Control, Vol AC-10, pp 250-254, Jul 1965

[SSH06] S. Schmid, M. Sifalakis, D. Hutchison. "Towards Autonomic Networks". In proceedings of 3rd Annual Conference on Autonomic Networking, Autonomic Communication Workshop (IFIP AN/WAC), Paris, France, September 25-29, 2006.

[ST01] James P.G. Sterbenz and Joseph D. Touch, ”High-Speed Networking: A Systematic Approach to High-Bandwidth Low-Latency Communication” , John Wiley, New York, 2001.

[WCZS92] Wakeman, J. Crowcroft, Z. Wang, and D. Sirovica, "Layering considered harmful", IEEE Network, January 1992, p. 7-16.

[WGT98] David J. Wetherall, John Guttag, and David L. Tennenhouse. “ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols”. In IEEE OPENARCH, April 1998.

APPENDIX A

List of cross-layer shared information metrics from the currently existing literature.

Metric	Source	Format Type	Access Type	Scope	Accuracy needed	Validity period	Collection Method
Operational state (on/off, up/down)	PHY LL NET TRANS	Boolean	R/W	Node Node Network Network	High	RT	Fixed State
Signal strength	PHY	% Ratio	R	Node	Medium	RT	Estimation
Bit-error rate (BER)	PHY	Scalar	R	Node	Medium	RT	Accounting
X-mit power	PHY	% Ratio	R/W	Node	High	RT	Fixed State
Signal-noise ratio (SNR)	PHY	% Ratio	R	Node	Medium	RT	Estimation
Coding/modulation strategy	PHY APP	Enum	R/W	Node	High	nRT	Fixed State
Throughput	LL NET TRANS APP	Scalar	R	Node Node Network Network	High	RT	Estimation
Delay/latency	LL NET TRANS APP	Scalar	R	Node Node Network Network	High	RT	Estimation
Jitter	LL NET TRANS APP	Scalar	R	Node Node Network Network	Hugh	RT	Estimation
QoS status / connection reliability metrics (packet loss, goodput)	PHY LL TRANS	Record	R	Network Network	Medium	nRT	Accounting
Neighborhood information	LL NET TRANS APP	List	R/(W)	Node Node Network Network	High	RT	Accounting
Retransmissions	LL TRANS APP	Scalar	R	Node Network Network	High Medium Medium	RT RT RT	Accounting
Frame/packet error rate	LL TRANS	Scalar	R	Node Network	High Medium	RT RT	Accounting

	APP			Network	Medium	RT	
Medium condition (collision status)	LL	Boolean	R	Node Network	High Low	RT nRT	Accounting
Buffer state	LL NET TRANS APP	% Ratio	R	Node Node Network Network	High	RT	Accounting
Buffer event	LL NET TRANS APP	Boolean	R/W	Node	Medium	RT	Event
Scheduling strategy	LL NET TRANS APP	Enum	R/W	Node Node Node Node	Medium	nRT	Fixed State
Handoff event	LL NET	Boolean	R/W	Node Network	High	RT nRT	Event
Packet salvaging ARQ/FEC notifications	LL TRANS	Scalar	R	Node Network	Low	nRT	Accounting
Services (MPLS, RSVP, Intserv)	NET	List	R	Network	High	RT	Fixed State
Service support (e.g. lookup) response time	NET APP	Scalar	R	Network	Medium	RT	Estimation
Interface characteristics (throughput, bandwidth)	PHY LL TRANS APP	Record	R/(W)	Node Node Network Network	High	nRT	Fixed State
Routing info (neighborhood, costs, distances)	NET	List	R	Network	High	nRT	Accounting
Routing Strategy (LS vs DV, and protocol ver)	NET	List	R/W	Network	High	nRT	Fixed State
Gateway	NET	Boolean	R/W	Network Node	High	RT	Fixed State
Round trip times (RTT)	LL TRANS	Scalar	R	Node Network	High High	RT RT	Accounting
Retransmission timer information	TRANS	Scalar	R	Network	High	RT	Accounting
Ack Notifications, Receive window information	LL TRANS	Scalar	R	Node Network	High	RT	Accounting
Congestion window information	TRANS	Scalar	R	Network	High	RT	Accounting
Connection MTU	LL TRANS	Scalar	R	Node Node	Low	nRT	Fixed State
Connection state (sequence information, active	LL TRANS	Enum	R	Node Network	High	RT	Accouting

options, status),							
QoS Requirements (Bandwidth, Delay, Loss)	LL NET TRANS APP	Record	R/W	Node Node Network Network	Medium	nRT	Fixed State
Quality metrics (as perceived by the application or user – totally adhoc?),	APP	Record	R	Network	Medium	RT	Accounting
Power status	DEV	% Ratio Scalar	R	Node Network Node	Medium	RT nRT RT	Accounting
Availability of memory	DEV	% Ratio Scalar	R	Node Network Node	Medium	RT nRT RT	Accounting
Processing load	DEV	% Ratio Scalar	R	Node Network Node	Medium	RT nRT RT	Accouinting