Objective FP7-ICT-2007-1-216041/D-4.4

The Network of the Future

Project 216041

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

D-4.4
In-Network Management System Demonstrator

Date of preparation: **10-06-13**
Start date of Project: **08-01-01**
Project Coordinator: **Henrik Abramowicz**
Ericsson AB

Revision: **1.0**
Duration: **10-06-30**

Document Properties[1]:

Document Number[2]:	FP7-ICT-2007-1-216041-4WARD / D-4.4
Document Title:	**In-Network Management System Demonstrator**
Document responsible:	Giorgio Nunzi (NEC)
Author(s)/editor(s):	Fabian Wolff (FhG) Alberto Gonzalez Prieto (KTH) Bioniko Tauhid Giorgio Nunzi Dominique Dudkowski (NEC) Christopher Foley (TSSG) Virgil Dobrota Bogdan Rus (TUCN) Rebecca Steinert Daniel Gillblad (SICS)
Target Dissemination Level[3]:	PU
Status of the Document:	Final
Version:	1.0

Revision History:

Revision	Date	Issued by	Description
1.0	13/06/2010	Giorgio Nunzi	First approved revision

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

Abstract:

WP4 proposes a novel approach of network management, called In-Network Management (INM). Its basic enabling concepts are decentralization, self-organization, and autonomy. The

[1] Input of Title, Date,Version, Target dissemination level, Status via *"File /Properties/Custom*" in the Word menu
[2] Format: FP7-ICT-2007-1-216041-4WARD /<Deliverable number>
Example: FP7-ICT-2007-1-216041-4WARD / D-1.1
[3] Dissemination level as defined in the EU Contract:
PU = Public
PP = Distribution limited to other programme participants
RE = Distribution to a group specified by the consortium
CO = Confidential, only allowed for members of the consortium

design of key INM functions has been reported in the previous D-4.3.

This document reports the results of the prototyping activity of WP4. Three scenarios are proposed: dynamic deployment of connectivity in emergency situations; integration of INM cross-layer metrics with adaption function of the Generic Path; optimization of virtualized resources. These scenarios have been identified as results of the joint activities between WP4 and the other Workpackages in 4WARD.

The scenarios have implemented through a selected set of INM functions. To offer a unified view of the operations in INM, we describe the organisation interface and the use of objectives for configuring the network and controlling properties of the INM algorithms. As experience of our implementation work, we provide details about specific INM algorithms: INM Runtime, Clustering, Anomaly Detection, Monitoring, cross-layer QoS. For these algorithms we describe a decomposition into functional modules and interaction diagrams.

Keywords:

Future Internet, in-network management, self-management, real-time management, scalable and robust management systems, architectural elements, situation awareness, self-adaptation, prototype.

Table of Contents

Executive Summary

WP4 proposes a novel approach of network management, called In-Network Management (INM). Its basic enabling concepts are decentralization, self-organization, and autonomy. The design of key INM functions has been reported in the previous D-4.3.

This document reports the results of the prototyping activity of WP4. The objectives are twofold. First, it provides an analysis of potential exploitation scenarios of INM in the future Internet. Second, it reports the experience of an integrated implementation of the INM algorithms and provides a refinement of the INM design with detailed information flows.

Three scenarios have been identified as results of the joint activities between WP4 and the other Workpackages in 4WARD. The first scenario is about dynamic deployment of connectivity in emergency situations, where the interoperability function of WP2 is integrated with the INM real-time monitoring. The second scenario considers integration of INM cross-layer metrics with adaption function of the Generic Path considered in WP5: network coding is considered in our implementation as adaptation function. The third scenario applies INM function to virtual networks as considered in WP3. Here INM objectives are used to configure adaptive actions in case of congestion and therefore support operators' business models. The document briefly discusses the benefits of the INM approach in these discussions, which can be summarized in three major items: reduction of integration effort, simplification of management operations and support for operator's business.

The scenarios have implemented through a selected set of INM functions. To offer a unified view of the operations in INM, we describe first the organisation interface and we show then its configuration for objectives as well as controlling properties of the INM algorithms. This interface allows also for composition of INM algorithms and construction of workflows at local level (e.g. anomaly detection and triggering of adaptation function) and at domain level (e.g. construction of aggregated key performance indicators).

As experience of our implementation work, we provide details about specific INM algorithms: INM Runtime, Clustering, Anomaly Detection, Monitoring, cross-layer QoS. For these algorithms we describe a decomposition into functional modules and interaction diagrams. These instruments show how the functions designed can be implemented and mapped to the message flow in accordance to the INM interfaces.

This deliverable is one of the two final deliverables of WP4 and is complementary to D-4.5, which reports evaluation studies.

Terminology

AD	Anomaly Detection
ATR	Available Transfer Rate
CLQ	Cross-Layer Quality of Service
DSL	Domain Specific Language
GAP	Generic Aggregation Protocol
GMP	Global Management Point
GP	Generic Path
KPI	Key Performance Indicator
INM	In-Network Management
MC	Management Capability
OWD	One Way Delay
QoE	Quality of Experience
QoS	Quality of Service
SE	Self-Managing Entity

1 Introduction

This document is one of the two final deliverables of 4WARD WP4 and reports about the implementation activity In-Network Management (INM). The scope of this document is therefore complementary to D-4.5, which reports on evaluation results through simulations and analysis.

The purpose of the implementation for INM is first of all to provide a proof-of-concept of the approach proposed by INM. This is used to prove the validity of the concepts introduced by INM and appreciate how networks can be operated through INM. The implementation focuses on integrating components together and validating them against realistic exploitation scenarios for the future Internet. For this purpose we performed this activity in conjunction with other WPs in 4WARD, with whom we built common use cases for implementation. In general, the scenarios show the need of advanced functions for self-management and are presented as case study to integrate INM capabilities.
Besides maintenance of specific network properties (e.g. congestion control, negotiation of service level agreements etc.), real-time operations and integration with operator's business are of paramount importance for effective network management. We therefore refine the concepts of organisational interface and management by objective, which are basic instruments for using INM capabilities. We look then at each specific INM capabilities to show concrete interactions in the use cases.
The remainder of this document is organized as follows. Section 2 describes the exploitation scenarios that are proposed for demonstrating the impact of INM. Section 3 explains the implementation and integration of the INM capabilities. Section 4 concludes with final remarks.

2 Exploitation Scenarios for In-Network Management

This section describes the exploitation scenarios for INM. First we provide a general discussion of the impact of INM in the future Internet. Afterwards, we define three scenarios that have been elaborated in conjunction with other workpackages in 4WARD.

2.1 Impact of INM in the deployment of the future Internet

Network management is not only about achieving the best network performances in a certain situation, but rather about controlling a set of diverse aspects in the network, where "best" shall be referred to target objectives dictated by an operator. The purpose of INM is to enlarge this set of aspects in the network, so that more aspects in the future Internet can be optimized through management techniques. For example, when we look only at simple requirements, like repetitive one-time configurations or low-scale periodic polling of network status, this can be achieved today through scripts. Instead, when we introduce additional aspects in operating networks, the performances of the management functions must be enhanced. When we need to reduce the delay in retrieving management information, then INM proposes a set of self-adapting algorithms for monitoring to keep maximum delay under control. When we consider costs related to integrating the management logic to the network function, then INM proposes a distributed architecture, more efficient with this respect than scripting.

The expected benefits of INM include therefore technical and non-technical characteristics in operating networks. Coarsely, they can be grouped along these lines:

- Reduction of integration effort
- Simplification of management operations
- Support for operator's business

A demonstrator comes as a valid instrument to bring to concreteness both the quantitative as well as the qualitative aspects underlying the INM design. In fact it provides a realistic set of use cases where the new instruments can prove to be very helpful, it gives a proof of concept implementation of the concepts designed and adds in experience for concrete exploitation plans. The objective is twofold. On one hand the INM concepts get value from a visualization of possible application into realistic deployments of the future Internet. On the other hand some of the INM interfaces are detailed at implementation level and therefore provide examples of mapping the INM design into running code.
In addition to this, a demonstrator serves as an instrument to guide integration of INM with the other architectures developed within the 4WARD project, and the scenarios have been in fact defined in conjunction with other WPs in the project. Despite the vertical span of the INM design, the demonstrator does not include integration with operators' backend interfaces, such as OSS, policy refinement software, billing system etc. In fact they are more related to business aspects of each operator than to technical features of the network infrastructure. Integration with these systems requires instruments related to policy mapping and the AutoI project [7] can for example used as solution.

We have been investigating a set of different exploitation scenarios in INM, mainly targeting integration with other WPs in 4WARD. To maximize the collaboration across the different scenarios and therefore reuse at most INM platform and components, we have been working under a common setting: the exploitation scenarios have been created on the basis of this common setting.

The common setting is shown in Figure 1 and is labelled "management of real-time services under changing conditions". It shows two major requirements in the demonstrator. The first one is the presence of at least one operator who is in charge of deploying advanced services: these can be emergency services, large-band video chats with stringent and guaranteed QoS, or a mix of different services. The second requirement is the need for advanced management instruments that can tackle the complexity in operating the network, both for the technical aspects related to network performances as well as non-technical aspects related to operator's objectives.

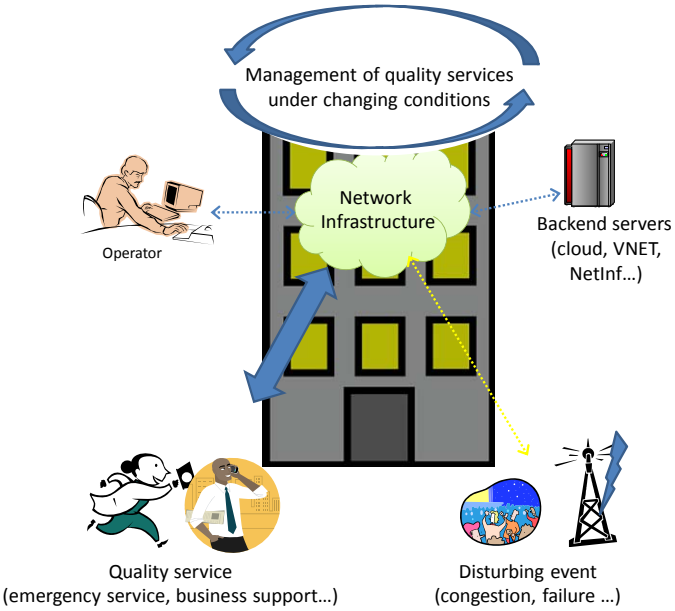


Figure 1 Reference setting for INM scenarios.

Given the reference setting above, we will show in the following scenarios that INM capabilities first of all automate management operations in two situations: they automate the configuration steps necessary to achieve connectivity at network level and integration at service level (similar to plug-and-play, but considering service aspects too); they enforce adaptive actions to maintain the required level of QoS and general objectives in the network.

In normal conditions, the network would behave like planned statistically by the operator and all the objectives would remain fairly well addressed without any intervention by the management function. The major challenges in the future Internet would instead stem from the occurrence of a disturbing event (failure, congestion etc.) The demonstrator leverages on the properties of the INM capabilities to build the necessary use cases of self-management, such as real-timeliness, autonomicity and local adaptation. The use cases are then implemented as reported in the remainder of this Chapter.

The expected results of these use cases are the following:

- Management operations are performed within a distributed architecture following the INM paradigm and their timeliness can be guaranteed over different scale of networks;
- Management operations are simplified in terms of configuration of high level objectives and aggregation of monitored data;
- New services, such as emergency services, can be supported in the future Internet quickly and in a reliable manner.

2.2 Real-time Autonomous Adaptation in Emergency Scenarios

This demo integrates the WP2 architectural constructs, i.e. strata, component based architecture, Service Level Agreement with the WP4 constructs, i.e. INM framework and algorithms. The prototype looks to demonstrate real time adaptation of networks in emergency scenarios.

The scenario addresses a natural disaster situation where emergency teams need to provide relief, but the network infrastructure is heavily damaged. Dynamicity and the need for timely adaptation are essential during or after these events. Figure 2 shows the domains involved in the scenario: (i) Fixed Operator (FO) Network, (ii) Emergency Medical Team (EMT) Network, (iii) Ad Hoc Disaster Recovery (AHDR) Network and (iv) Emergency Medical Teams.

The sequence of events is as follows;

1. FO exists and disaster occurs.
2. Emergency Medical Teams are deployed and communicate back through FO to Emergency Response Command – SLA negotiated and agreed.
3. Ad-hoc Disaster Recovery network is deployed and bootstrapped
4. Real time monitoring of network congestion is switched on in FO – using the GAP algorithm
5. SLA Module for FO subscribes to INM for congestion levels
6. Congestion levels in FO are forced upwards(simulating the impact of the disaster) – this is visible through GAP(Ref to section 5.6)
7. GAP triggers self adaptation, by invoking the Congestion Control algorithm
8. Congestion control attempts to self adapt to balance load inside FO. This is achieved by moving traffic onto a less congested path which balances the overall network congestion level.
9. Congestion levels are forced upwards again, this time to greater than 80% network congestion.
10. The congestion level goes over the threshold set by the SLA – SLA renegotiation with SLA of the AHDR.
11. This results in one of the emergency team being transferred over to use the AHDR. Congestion in turn decreases in FO and traffic increases in AHDR.

The challenges in this scenario are related to the establishment and maintenance of a good Quality of Service to guarantee a reliable connection to the emergency team. The major limitation is that instruments for dynamic reconfiguration are not put in place in the current networks. Integration of the WP2 and WP4 constructs allows for easy deployment of networks, real-time monitoring and self adaptation to occur within network domains and also interoperability between network domains.

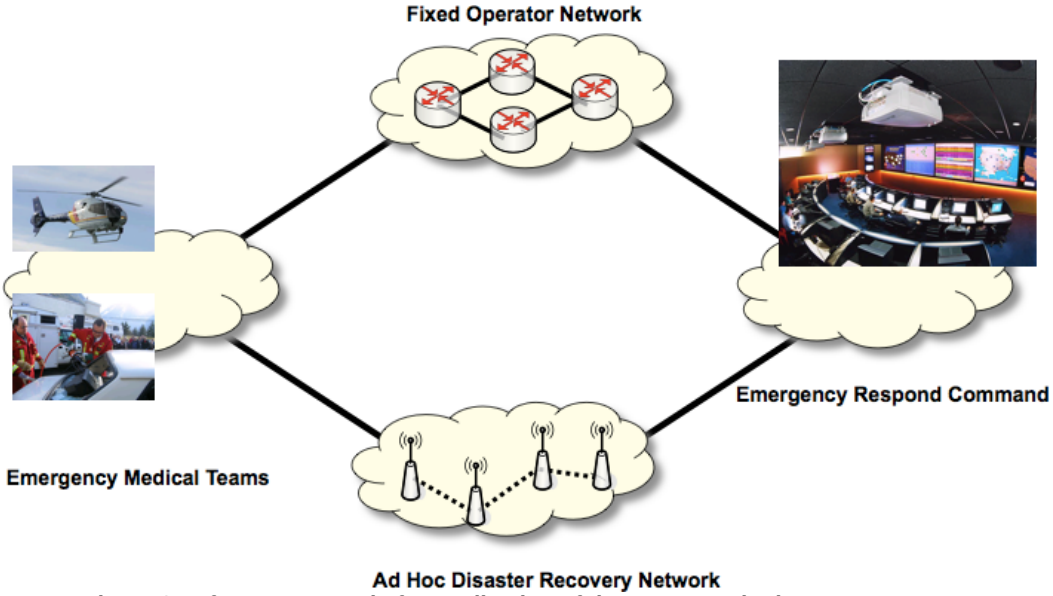


Figure 2 Reference scenario for application of the prototype in the Future Internet

The GAP algorithm, realised through a number of Management Capabilities (MCs) is deployed in the FO network. This is monitoring the overall congestion level of the network and publishing this as an INM Objective (technical details are defined in section 3.2). All INM objectives can be seen through the INM Console. The Congestion Control algorithm is also deployed inside the FO network but is not active from the start. As congestion increases GAP monitors it (see D-4.3 [1] for more details) and publishes via the INM Framework. The Knowledge Strata for the FO network subscribes to the objective which GAP produces. This objective represents knowledge produced by the system. In essence the INM monitoring algorithms are knowledge sources. When congestion reaches 60% in FO network, there is interaction between the Knowledge and Governance strata (see D-2.3 [3] for more details). The Governance stratum triggers the Congestion Control into action, i.e. to try and resolve the congestion by using load balancing on different paths within the network. Figure 3 gives an architectural perspective on the integration of the WP2 and WP4 components used in the demo.

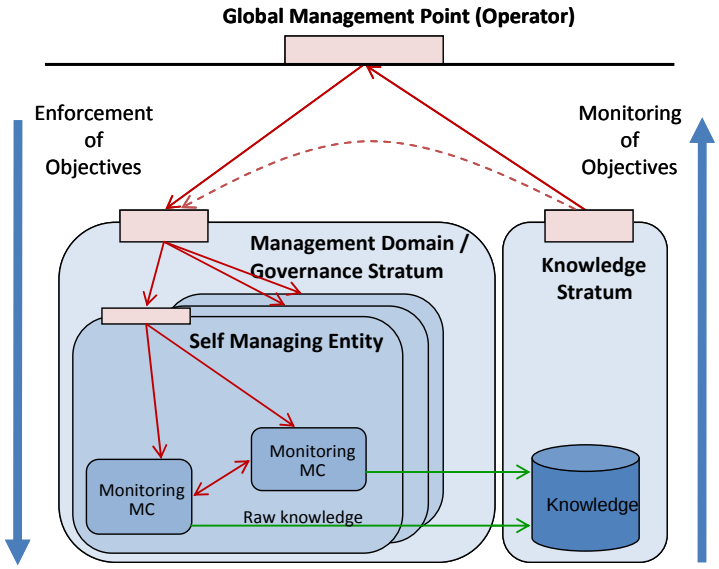


Figure 3 Integration of Management Capabilities (MC) and Strata

The congestion control algorithm through routing strategies handles the congestion within the FO network.
To trigger network interoperability functionality a dramatic increase in congestion (>85%) build up is simulated inside the FO network. With this the Governance strata acts on the Knowledge strata and informs the “SLA Manager” of the congestion, and as there is a SLA violation the “SLA Manager” in the EMT network decides to switch the service level agreement from the FO domain to the AHDR domain. Congestion is then reduced inside the FO network and traffic increases in the AHDR network.

In this prototype implementation, the INM framework provides the minimal functions common among INM algorithms to support self-organization: a naming scheme for INM algorithms, discovery, organisation interface for control of objectives and a visual interface for operators. From a functional point of view, the prototype integrates real-time monitoring of network-wide metrics and local triggering of self-optimization: these functions are applied to the case of management of dynamic traffic fluctuations in the network and healing of congestion.

Thanks to the gossiping protocol “INM Monitoring” a network wide congestion status can be estimated, while preserving resource consumption. In case of high network congestion (i.e. a pre-configured congestion threshold is crossed), “INM Monitoring” via strata autonomously invokes the INM algorithm “Congestion Control”. This has the resulting impact of having network congestion efficiently distributed and balanced via multiple possible routes and if needed the congestion status of each router and available path can be visualized in a detailed view.

The INM framework and algorithms for monitoring and congestion control provide distributed domain level congestion detections with optional self-healing capabilities while also allowing domain owners manage their domains at an objective level.

2.3 Cross-Layer QoS and Generic Path for Real-Time Congestion Control

Congestion is believed to be nowadays a major problem in data communication networks, because it has a major influence over the quality of experience (QoE) that characterizes the

services received by a user. A possible solution to this unwanted situation would be to reroute the traffic through a higher transfer rate link. Even though this method seems to be the most suitable one, it is not always realistic because such a route might not be available at that moment. In the demonstrator presented herein we propose another approach that deals with the existing congestion (without eliminating it) while preserving the quality of the provided services.

The INM prototype can show the advanced functionalities of the transport plan in the future Internet through an integrated self-management use case of Generic Paths. A typical motivational case (shown in Figure 4) is the co-existence of diverse traffic sources in the network, such as those generated by two service providers (YouTube and Metacafe), one transport provider that is 4WARD-aware and two users (John and Mary). User Mary is a Metacafe's subscriber, whilst John is requesting audio & video streams from YouTube. Both services are being delivered through the same transport provider.

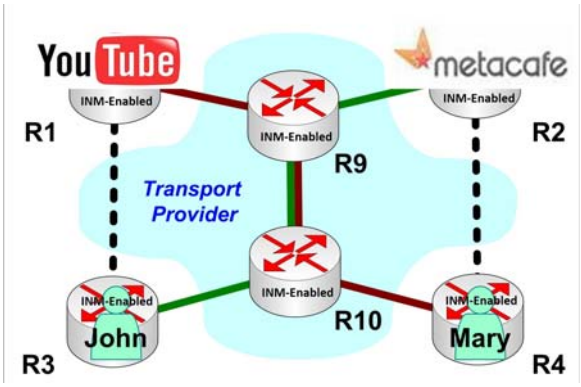


Figure 4 Case 1: no congestion in a non GP-aware network

For a clear view of the benefits that the demonstrator brings, three cases are considered as it is depicted in Figure 5. The testbed considered is a butterfly topology which is complex enough to allow NC-based GP [4] but it is simple enough to allow real implementation.

INM Cross-Layer QoS	Congested link	GP-aware
✓	X	X
✓	✓	X
✓	✓	✓

Figure 5 Three cases of the scenario

At a given moment congestion on a shared link occurs (Figure 6) and it is detected by the monitoring component of the INM Cross-Layer QoS (CLQ) management capability. In the scenario considered, no additional resources are available so that CLQ triggers the instantiation of a Network Coding (NC) - based Generic Path (GP) on a six-node butterfly topology, performing a distributed enhanced routing in order to face congestion (Figure 7). Statistics based on average available transfer rates measured on the links are the ones that trigger the activation/deactivation of this 4WARD congestion control mechanism.

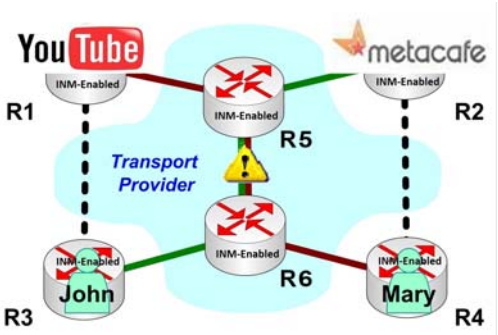


Figure 6 Case 2: congestion in a non GP-aware network

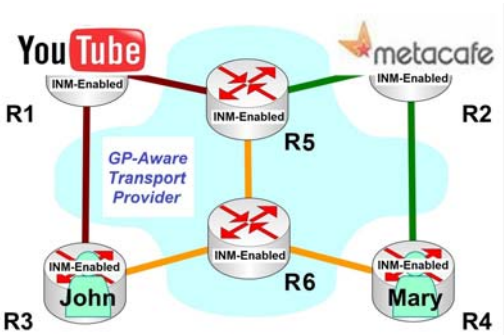


Figure 7 Case 3: congestion in a GP-aware network

The major **immediate benefits** that the demonstrator shows are the following:

- A better use of limited resources when cooperative networks are involved being an alternative solution to complex QoS-aware routing.
- The feasibility of 4WARD principles regarding the usage of CLQ management capability for congestion control, as a module that activates the instantiation of a NC-based GP

With respect to the **potential benefits** that can be envisaged when combining a management capability like CLQ with a NC-based GP the following exploitation scenarios can be taken as examples:

- wireless backhaul network in rural areas (Figure 8)
- generalisation of a network as being composed of butterfly cells (Figure 9).

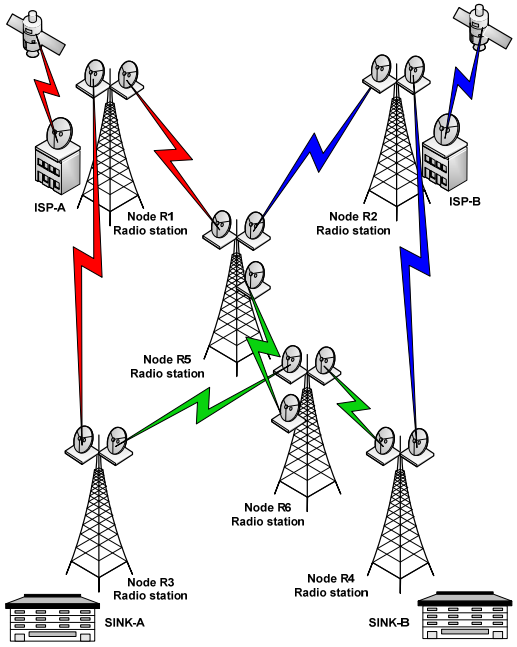


Figure 8 Potential benefits is wireless backhaul network in rural areas

In wireless backhaul networks deployed in rural areas (Figure 8), link transfer rates are always limited. Because the transmission channels are presenting dynamic characteristics (i.e. variable attenuation) the system is suitable for implementing an adaptive GP. This activates

the coding mechanism only when needed, being triggered by the CLQ capability. The intermittent activation of the coding mechanism is mandatory when we want to avoid the unnecessary engagement of resources (CPU, memory, bandwidth etc.).
The second potential benefit is presented in Figure 9 where can be seen that a butterfly topology can be easily found in real-life networks (actually the network is composed by a set of overlapping butterfly cells) so that the NC-based GP is far from being a solution for particular and topologies. Note that MP is a Mediation Point and EP is an End Point.

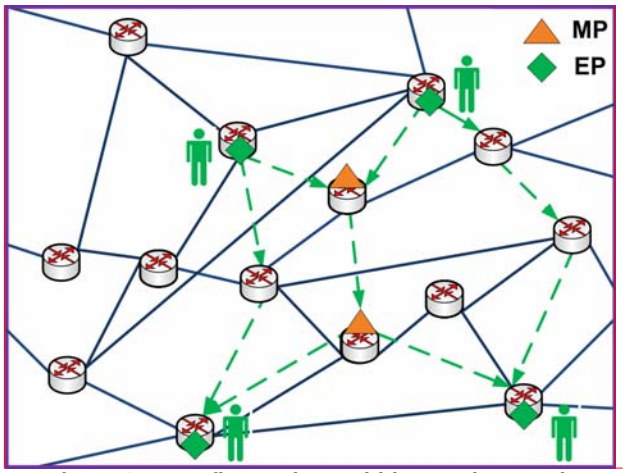


Figure 9 Butterfly topology within a real network

Comment [GN1]: What is MP and EP??

The complete architecture of the testbed, containing both the CLQ and the NC-based GP mechanisms is depicted in Figure 10. It can be observed that in every node, the CLQ capability accesses the hardware and monitors the situation in the network. When congestion is detected, the signalling mechanism triggers the NC-based GP, activating the necessary operations into the mediation points. The topology consists of multiple sub-GPs, connected through MPs. The architecture includes: multicast sub-GPs (that are transmitting the same information on every link), and coded sub-GP (that transmits coded information received by multiple sources).

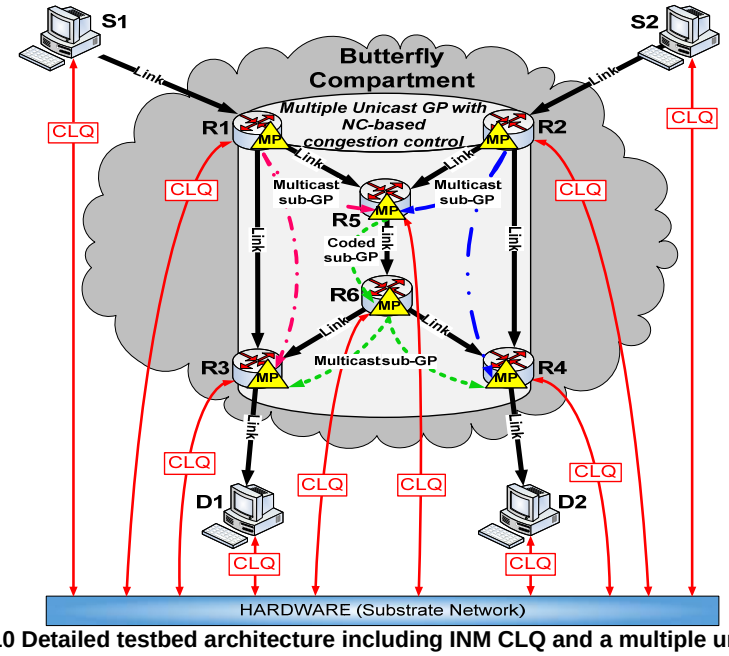


Figure 10 Detailed testbed architecture including INM CLQ and a multiple unicast GP

The whole interaction mechanism of the tesbed is illustrated in Figure 11. The main components are: the INM CLQ capability, the local MySQL database, the statistics mechanism and the NC-based GP. The way that these components work together is the following: the CLQ capability monitors using cross-layer techniques the performance indicators of the network (One Way Delay – OWD and Available Transfer Rate – ATR). These parameters are inserted into the local database and also published to the rest of the nodes in the domain so that the information of every link in the network is stored into the local DB. The statistics module is querying the parameters from the local DB in order to trigger the NC-based GP activation. The signalling between CLQ and GP is made through a text file named statistics.txt. More details are presented in paragraph 5.4.

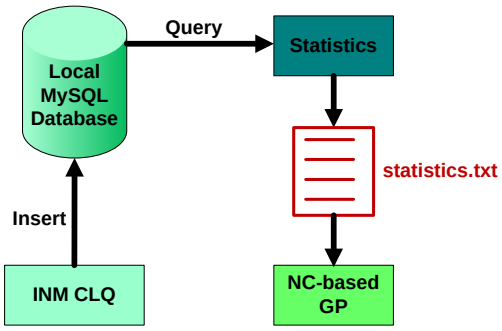


Figure 11 Processing scheme

2.4 Simplified management in virtualized environments

Virtualized networks introduce some additional complexity in management operations, because they rely on two particular technical enhancements: first connectivity is maintained across different networks (different in ownership or technology, e.g. wireless/wired), second connectivity is built at different layers (namely physical and virtual layers).

The aspects above can be seen as additional requirements in terms of scalability, because management functions must be enforced and integrated across the different sectors (vertical across layers or horizontal across technologies) of a virtual network.

An example deployment scenario for In-Network Management of VNETs is shown in Figure 12. Here, two virtual networks are deployed over a shared infrastructure, they compete for resources over certain nodes that are shared, and they are operated by different operators that are competing to attract customers and to differentiate their service offers. The management of the infrastructure implies a maximization of utilization of the resources, in particular those shared across the different

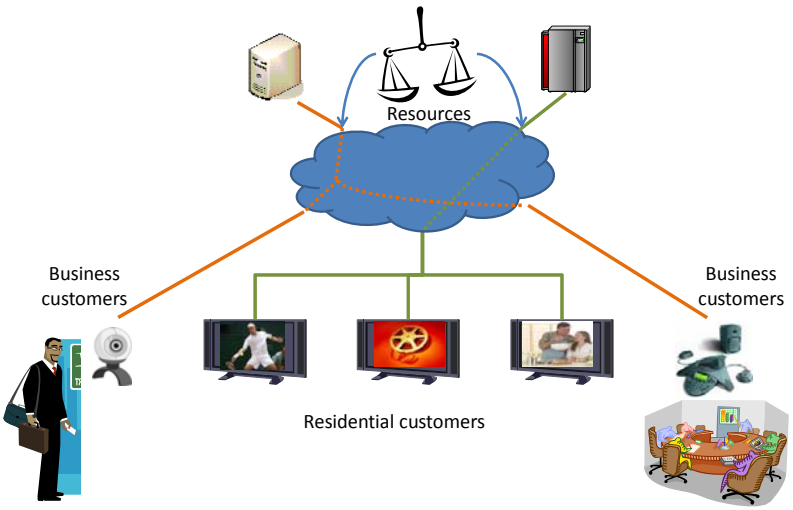


Figure 12 Deployment scenario of INM applied to virtualized networks.

In general it can be assumed that resources are carefully planned and allocated following the deployment workflow deployed by WP3, but an underlying sharing and multiplexing of resource across different VNETs is implied to assure efficient resource usage and cost saving with respect of traditional networks. The challenges are twofold: on one hand, certain key performance indicators must be maintained network wide on each VNET (for example a certain amount of maximum throughput); on the other hand, the performances of certain network functions must be controlled locally in more stringent time and continuously optimized if necessary (e.g. delay between points of presence in the VNET).

INM provides a structured architecture to manage virtualized networks across different topologies and operations. Management operations are enforced through a coordination of self-managed objectives at different scales: domain-wide objectives, which monitor aggregated values, and local objectives, which monitor local performances of a network function. These objectives and their related INM capabilities permit to have control of the entire physical infrastructure for VNETs as well as to optimize resources accordingly to the required service level agreements.

The coordination of objectives in different parts of the network enables an abstraction of much of the complexity in managing single network elements, and this approach can be effectively applied to VNETs for increasing scalability of their management operations. This application can be pursued through three steps:

1. Definition of INM objectives for a specific network domain. A wireless network needs different relevant performance indicators than a wired network; additionally, performance on the virtual links must be measured differently from physical links.

2. Coordination of management information across different domains. A congestion on a bottleneck can cause bad performances in different areas of the networks. In future networks the identification of a root cause on different areas of a large network will reach particular relevance also due to the increased available bandwidth of the infrastructure.

3. Generation of aggregated management information, also by suppressing superfluous information in one domain.

Error! Reference source not found. shows the coordination between physical and virtual and an example applied to congestion control. We envisage a specific interface to communicate between the two layers, which is implemented by a Generic Management Point. A critical aspect in managing virtual networks is to keep transparency between the two layers, i.e. not all the information can be exchanged. The GMP acts therefore as a filter, exposing to the virtualized only basic management information, like presence of exceptions or status of self-adaptation in the substrate. The right part of **Error! Reference source not found.** shows an example applied to congestion. Here the INM functions in the physical layer detect the anomaly and the exception is forwarded to the GMP as notification. There is no risk of overloading the virtualized resources with fault notifications, because congestion detection is already aggregated. The functions monitoring on the virtualized resources can therefore receive real-time notification of self-management in the substrate. Coordination between the two layers is therefore guaranteed.

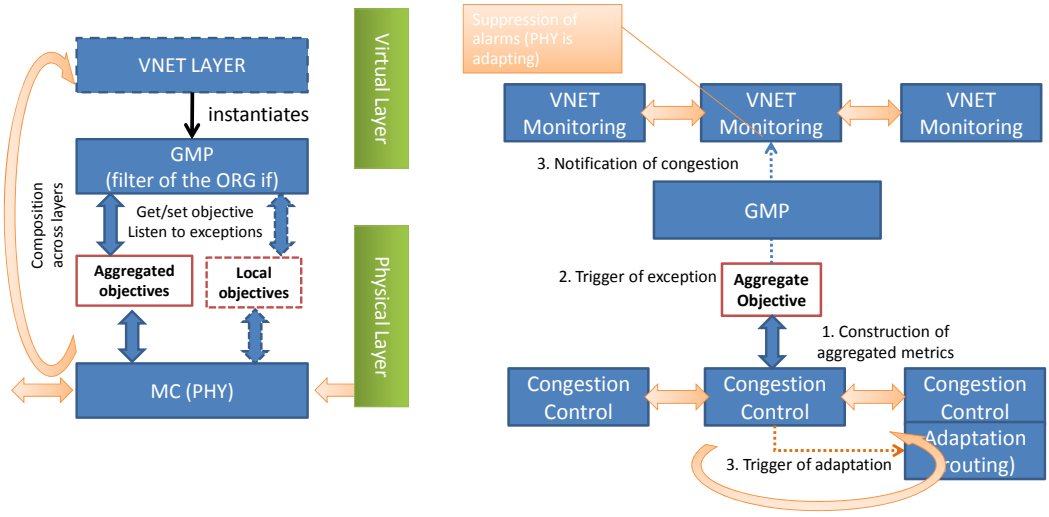


Figure 13 Coordination between physical and virtual layer (left) and congestion control use case

3 The INM Integrated Prototype

This section provides detailed specification of a selected set of INM algorithms and functionalities of the INM framework used for the integrated prototype. The prototype has

been used for the different demonstrative scenarios described in the previous Chapter, and therefore some of the examples are based on concrete use cases.

3.1 INM Runtime

The INM Runtime provides a set of basic functionalities to deploy INM capabilities and to support communication among them.

Objectives for network management

INM supports scalability through the introduction of a simple set of information elements to describe network and to disseminate status and decision across different elements. These information elements are designed for distributed algorithms and as instruments to build aggregated interfaces for management operations. The tight use of the information elements of the INM framework and the distributed algorithms enables the enhanced timeliness and scalability in INM.

The definition of an objective is shown in Figure 14: it defines the expected behaviour of a network function and its deviation from the actual value. The behaviour can be defined through different value types (e.g. integer). An objective might have a target associated to it, or might not. An objective might have different scopes.

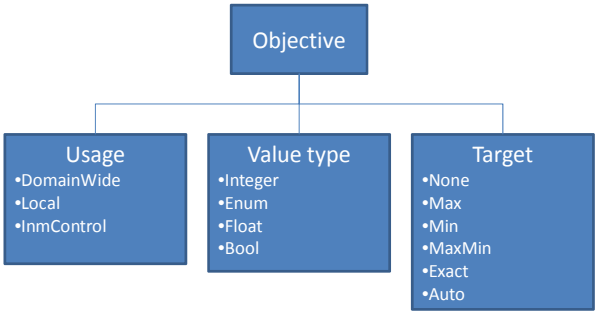


Figure 14 Definition of INM Objectives (design phase)

INM algorithms work at different levels of aggregation, and therefore objectives are associated with a “usage”, which allows them to self-organize objective across INM algorithms distributed in the network. The possible usages of an INM objective are:

A. Local: this usage specifies that this objective affects a single node and controls the behaviour of a network function. For example, detection of anomalies on a single link can be enforced with the definition of a Local objective.

B. DomainWide: this usage specifies that this objective affects a set of nodes and controls the behaviour of a network function on those nodes. For example, real-time monitoring of number of flows in an entire domain can be enforced with the definition of a DomainWide objective.

C. InmControl: this usage specifies that this objective controls the INM algorithms. D-4.3 showed that INM algorithms are in general designed with tuneable properties (e.g. performance, accuracy etc.): these properties cannot be adjusted internally by the algorithm, but they must be derived from external factors such network technology, cost factors etc. Objectives of usage InmControl can be used to enforce these external factors into the behaviour of the distributed algorithms in INM.

The concept of domain has been introduced in D-4.3 to restrict the scope in which INM algorithms compose between themselves. A domain can be used to specify the set of nodes where aggregation takes places. Aggregated objectives are therefore applied to those nodes.

Example1: GAP monitors network-wide aggregates. When coupling GAP with GP to monitor the number of flows, then an objective with the DomainWide scope would be created in the implementation. The objective would be something like <objective-id=gp:numberFlows, objective-scope: DomainWide>.

Example2: AnomalyDetection monitors local link anomalies. The objective implemented would be something like <objective-id:anomaly,objective-scope:local>

Example3: GAP and AnomalyDetection introduce certain behavioural p arameters, like performance, accuracy or cost. An example of such objectives would be <objective-id=gap:accuracy, objective-scope: inmControl>

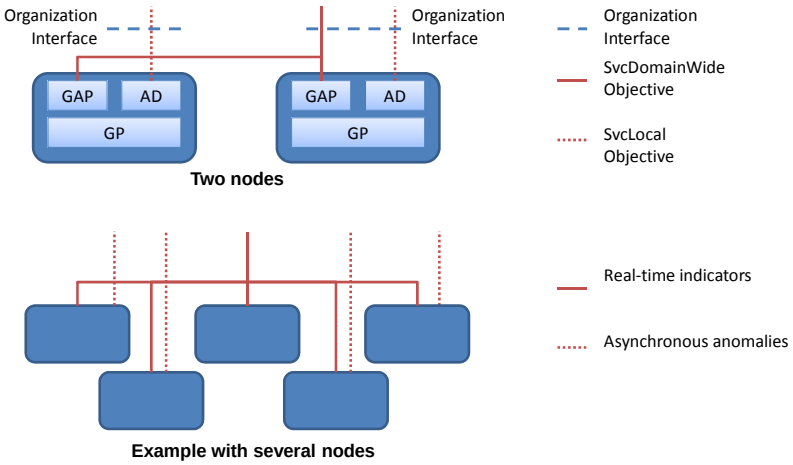


Figure 15 Usage of objectives and an example with several nodes.

The Value type is similar to traditional languages for information modelling. An objective needs to be mapped to one of the well-know value types in network management (integer, enum, Boolean etc.) Additionally this definition can ease the integration into legacy management interfaces on the backend (one-to-one mapping to existing code).

Example: AnomalyDetection defines a set of alarms that can be mapped to a set of enums.

The Target defines whether the objective is self-managed (only exceptions are generated) or not (actual values of the objective are generated real-time, like a key performance indicator). The types of targets can be mapped to one of the well-known logic statements used in threshold detection (e.g. max/min/both threshold, exact value etc.).

Example: the objective <objective-id=gp:numberFlows, objective-scope: DomainWide> can have associated a target, stating the maximum number of flows. The complete definition of the objective would then be <objective-id=gp:numberFlows, objective-scope: DomainWide, objective-target:max>

Organization Interface

The organization interface is used mainly to access the objectives during run-time. INM distinguishes at run-time TWO values of an objective: (i) the actual value and (ii) the target

value. Clearly the target value can exist at run-time only if the definition of the objective is self-managed. If the target value does not exist, the objective is only used for construction of a performance indicator.

The information elements accessible through the organization interface are two. The actual value represents the real-time value of an objective. The timeliness of the actual value is a tuneable property in INM and it is controlled. The target value is instead the expected value on an objective. The target value is extracted from operator's objectives, for example through refinement of high-level objectives and enforced through the organization interface.

The access to the objectives requires specific commands to read/write as well as to support asynchronous communication between INM capabilities (see Figure 16). The objectives can therefore be controlled through two mechanisms:

- Synchronous access. It is used for access of the actual and target values and for discovery of the objectives. It is defined through the Organization Interface and is composed of the following methods:
  - getObjectives: returns a list of the objectives defined by a Management Capbility (MC). The list is defined through a list of IDs.
  - getActualValue: returns the actual value of an objective. The objective must be specified through its name. The value is specified through objects oriented techniques: for example in an implementation in Java a common high level object is used for a generic INM objective, which needs then to be further refined. Note that when a target is defined, the actual value can be represented as a simple red/green status.
  - setTargetValue: sets the target value of an objective. The objective must be specified through its name. The format of the value follows the same indications as above. In addition, the operation must specify the type of target. If the type of target is not supported, then an error is returned.
- Asynchronous access. It is used for exceptions, triggers and announcements. Events are further explained below in this sub-section.

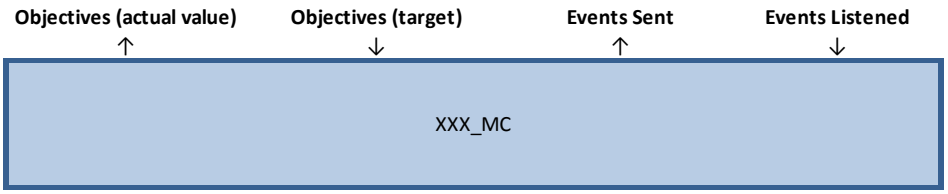


Figure 16 Operations on INM objectives

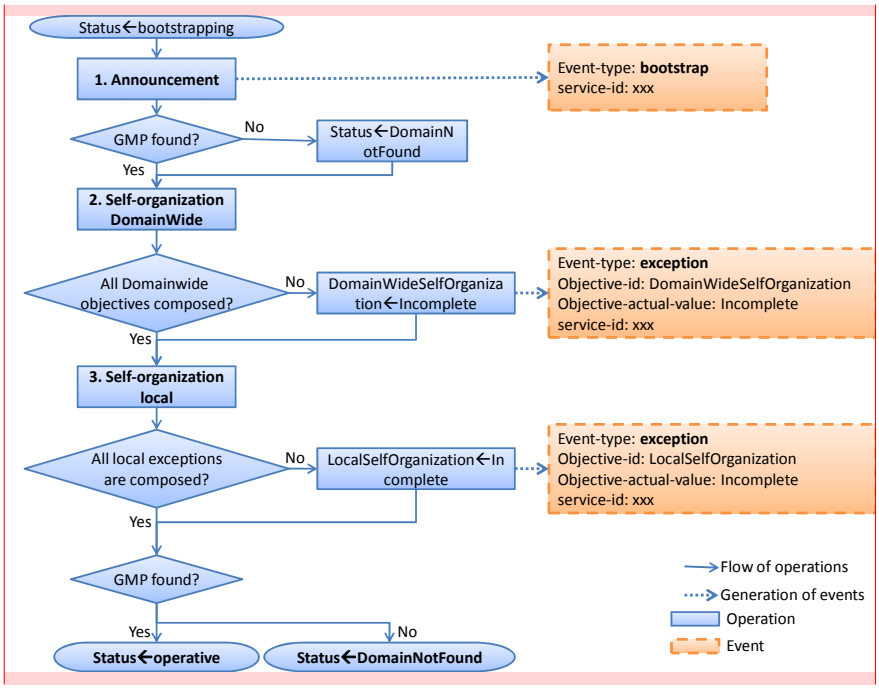
Generation of events for triggers/exceptions

Type of events:

- Triggers
- Local Exceptions
- DomainWide Exceptions

Es example, we consider the bootstrap phase of a capability. The purpose of the bootstrap is to assure that the objectives are discovered, properly self-organized and their exceptions properly handled. This happens in the following phases:

1. Announcement. A GMP is advised of the bootstrap phase.
2. Self-organization of domain-wide objectives. All domain-wide objectives are composed within the domain. At the end the objective “DomainWideSelfOrganization” is set.
3. Self-organization of local objectives. All local objectives are tested to verify that self-optimization can take place. At the end the objective “LocalSelfOrganization” is set.
4. Final announcement. The GMP is advised of the success of the bootstrap (not shown in figure).



Comment [GN2]: Need to revise it

Figure 17 MC bootstrap workflow

3.2 Workflows for In-Network Management

The usage of the INM objectives support the enhanced properties of algorithms, namely scalability, real-time monitoring, fast reconfiguration as well as cost-efficiency of the entire infrastructure. Then who is using the INM objectives and how are they mapped to operators’ business?

The first step is to clarify the distinction between InmControl objectives and objectives with DomainWide/Local scopes.

Distinction between InmControl and DomainWide/Local objectives

The previous section 3.1 has introduced a categorization of the INM objectives: their usage is here described more in detail. InmControl objectives can be used to control the degree of autonomicity and distribution of INM algorithms. For example, if a network is planned through over-provisioning principle, then some advanced self-management functions (e.g. congestion control) can be disabled; if the network is planned for maximum utilization, then self-optimization would be activated and monitoring algorithm would be configured for more effective operating mode. Practically, a network architect would estimate some high-level objectives in terms of degree of distribution and autonomicity; these high-level objectives are

then refined to InmObjectives through a Domain Specific Language (DSL, see section 3.2.2) or solutions based on ontology (see AutoI project [7]).

DomainWide and Local objectives can be used to generate Key Perfomance Indicators, to trigger self-optimization or to report exceptions. The definition of this information is in general specific to each operator. The distinction of the two categories of INM objectives is depicted in Figure 18.

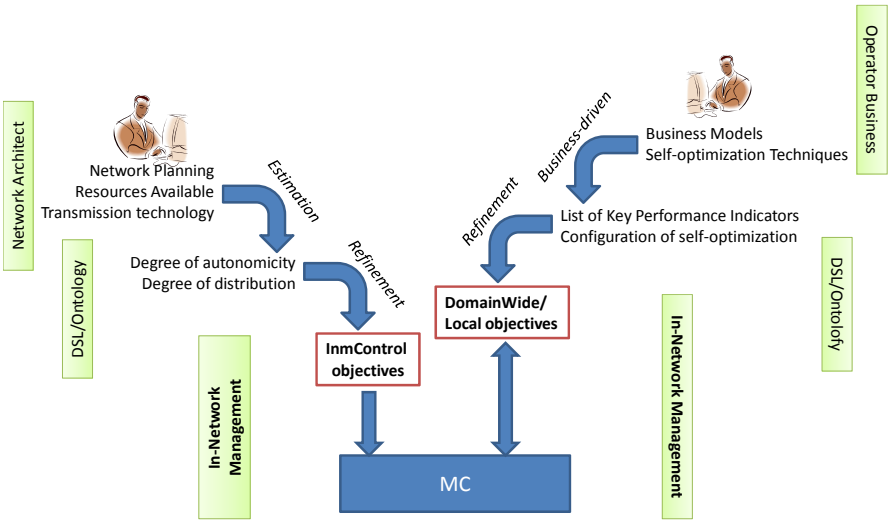


Figure 18 Distinction between InmControl and DomainWide/Local Objectives.

Distinction between DomainWide and Local objectives
DomainWide objectives are mainly used for producing real-time objectives of the network. The information reported is necessarily aggregated across different network elements and is presented in terms of KPIs (e.g. built through GAP) or status information (e.g. healthy conditions of an entire domain). In addition, NeworkWide objectives can also generate exceptions, for example when a MaxTarget is associated (similar to traditional threshold detection, but on a large scale).

Local objectives are mainly used to trigger self-optimization, which necessarily is performed over a selected set of network elements (e.g. anomaly detection triggers re-rerouting over the set of network elements affected by the anomaly). A local objective can also generate an exception; this can happen when the self-optimization did not bring the local objective to the target value.

From the analysis above, it appears evident that also exceptions must be distinguished on the basis of their scope. If local objectives would report directly an exception, then event storming would occur. Local exceptions must therefore be aggregated as well.

The actual point of implementations of the DomainWide objectives is the General Management Point, which is implemented by one MC in the domain.

The relation between the objectives at different scales is depicted in Figure 19.

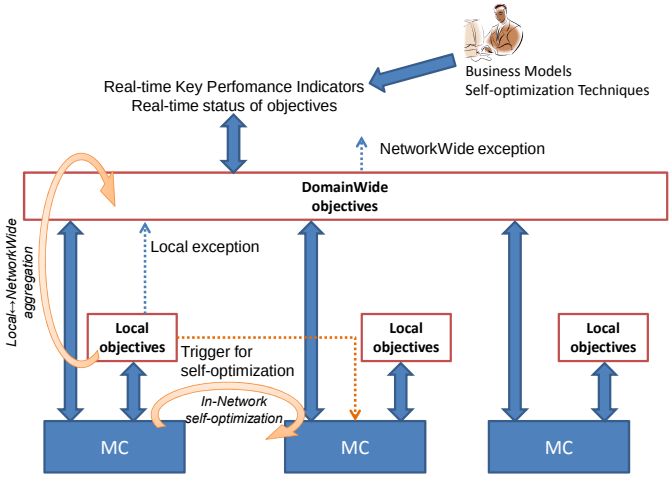


Figure 19 Distinction between local and DomainWide objectives.

3.2.1 Creation of Domain Wide Objectives

The deployment of algorithms results in one or more DomainWide objectives becoming visible in the system. As part of algorithm development DomainWide objectives are created.

As Figure 18 shows, there is the need to define a set of composite objectives that map operator's requirements to the logic of low-level INM objectives. For this purpose, we propose here an additional mechanism for the creation of dynamic objectives that is using a Domain Specific Language (DSL). DSL is a computer language that targets a particular kind of problem, rather than a general purpose language that's aimed at any kind of software problem.

The DSL developed for the INM framework is specifically formulated for composing multitudes of existing running DomainWide objectives. So what is created using the DSL is a 'Composite DomainWide Objective' (CDwO).

The notation of the DSL is easily readable and understandable by domain experts (e.g. Network Architects, Business Analysts). This offers obvious benefits in verifying correctness, maintenance and overall flexibility and accuracy of the solution.

Figure 20 shows the main components and high level interactions involved in the creation of a Composite Domain wide Objective. A user or administration creates a DSL script to generate or update a Composite Objective and deploy it in real time environment. A CDwO can access an existing objective in the network. It can subscribe to any number of objectives (domain wide or local) and perform operations/comparisons on the results which are returned by the objectives. It also has the possibility to take actions based on what it receives e.g., if an actual value is greater than 80%, then trigger alarm.

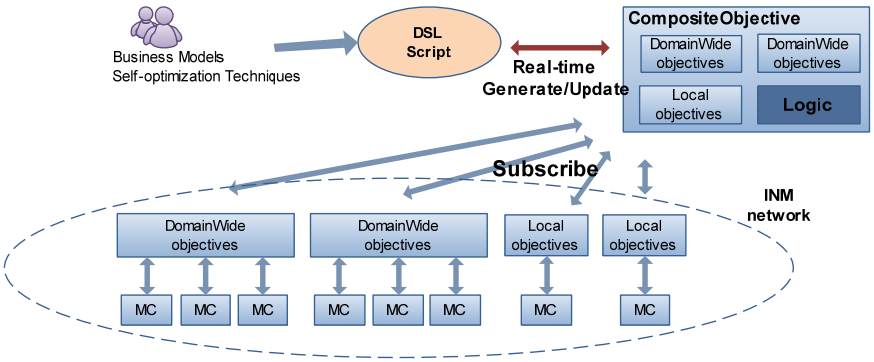


Figure 20 Composite Domain Wide Objective Creation via DSL

The CDwO workflow (Figure 21) divides in four stages.

1. **Initial** - Check the composite objective behaviours. There are two ways to access the actual value or deployed objectives synchronously or asynchronously (events). The synchronous approach is to directly request a value of an objective from the MC through the Organisation Interface. The asynchronous approach is to subscribe to that objectives and receive events when there are changes to the values of that objective.
2. **Waiting** – the script waits for the values to be returned and when received then evaluates them with respect to whatever condition exists.
3. **Trigger** - an action may be triggered if the condition(s) evaluate to true, otherwise it will go back to the waiting state and wait for the next update notification.
4. **Check Status** - The status is checked in each round to make sure the request is completed and should terminate or not.

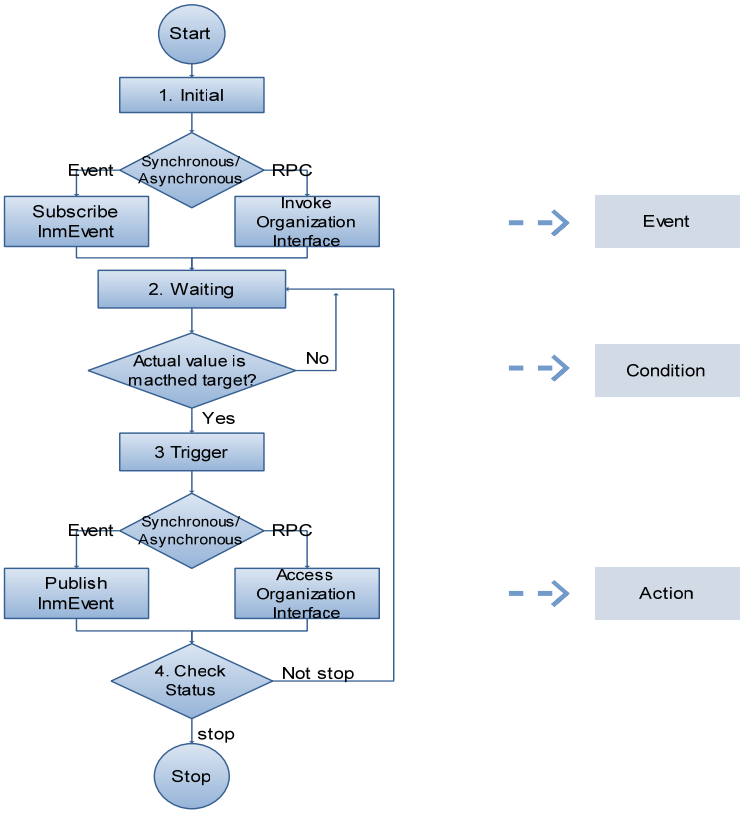


Figure 21 Workflow for composite objectives

3.2.2 DSL Design

A Composite Objective contains a set of rules. Each of Action associated with a condition is Rule as shown Figure 22 (A). Each condition has set of parameters. The parameter can be an event or anther condition result as shown Figure 22 (B). The parameters are described in detail in the following.

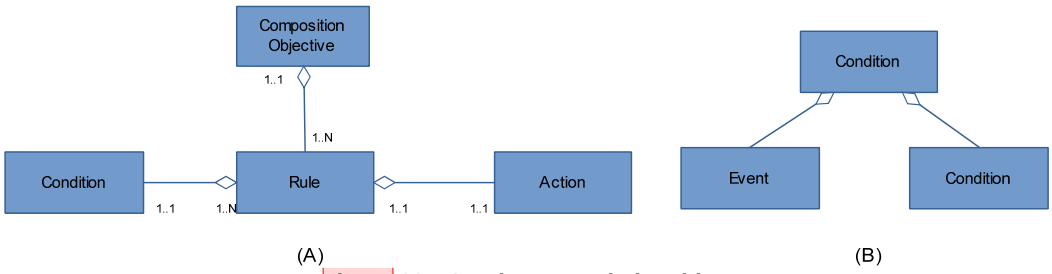


Figure 22 DSL Element Relationships

Comment [GN3]: Better description?

Event: used to create a subscription, each published event has set of key and value pair attributes for user to subscribe. DSL syntax shown below:

```
createEvent(       - feature name
      commonName   - subscription name of Event(id)
      status       - method name(invoke or subscribe)
      properties   - key and value pair
      keyValue     - actual interesting value
      )            - end
```

Condition: it is set of logic. The condition consists of parameters. For example, “if” condition has three parameters, which are a logic symbol and two comparison values. DSL syntax shown below;

```
createCondition( - feature name

      commonName – name of condition(id)
      operation  - method name(if or calculation)
      para       - value for method
      )          - end
```

Action: it can be a simple instruction that may trigger exception, such as publish an event or access an interface. DSL syntax shown below:

```
createAction(     - feature name
      commonName – name of Action(id)
      status      - method name(publish or trigger invoke)
      destination- actual interesting value
      objective   - key and value pair attributes
      )           - end
```

Rule: it is the link between condition and action, each action is associated with only one condition,. DSL syntax shown below:

```
createRule(        - feature name
      commonName   – name of Rule(id)
      condition    – name of condition(id)
      action       – name of action(id)
      flag         - optional for action, only invoke once
      )
```

The overall Composite Objective is made up of Events, Conditions, Actions which are all encapsulated into a Rule. DSL syntax is shown below:

```
build.task(){           - start create task
createEvent(commonName:"eventID_1",… )
createCondition(commonName:"c1",…)
createAction(commonName:"a1",…)
createRule(commonName:"rule1",condition:"c1",action:"a1")
}
return build.CompositeObjective("comp1") – Named Composite
                                          Objective and return
```

3.3 INM Clustering

The INM clustering is a capability that supervises the execution of distributed INM algorithms within a domain. It is responsible to monitor the exceptions raised within the domain and to report unhandled exceptions that must be forwarded to the operator for attention. In other words, this capability introduces some kind of coordination between INM algorithms.

This capability introduces a certain level of structure in the network and acts as master within a domain. One might argue whether this capability introduces a bottleneck to the execution of INM algorithms, but it should be noted that scalability of the entire self-management is not affected for two reasons. First, the amount of information that is passed to the clustering capability is very low: it mainly handles exceptions and the amount of information generated at this level is certainly a fraction compared to the entire real-time messaging in INM. Second, the purpose of this capability is to make the self-management behaviour of the INM algorithm visible to the operator. Only exceptions that are not handled or recovered in time are forwarded to the operator. Nevertheless this does not affect the normal operations of the single capabilities, which are therefore executed normally without interruption.

The INM clustering implements a state machine to monitor the activity of the INM algorithms, shown in Figure 23. It differentiates between different states in the network, and state changes are triggered by events coming from monitoring and adaptation functions of INM. A timeout is also enforced to guarantee that a timeliness requirement is enforced. This timeout is regulated as an objective from the operator and is aligned with the timeliness parameter of the INM algorithms. For example, if the timeliness of the INM algorithms is set to 100ms, a similar range is enforced as timeout.

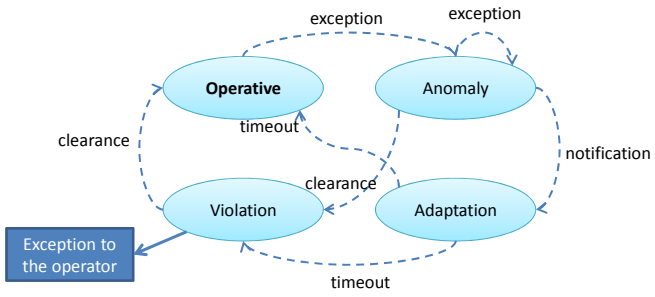


Figure 23 The state diagram implemented by the clustering capability.

The clustering capability allows to trace and to coordinate the actions of the INM algorithm. It is responsible also to identify the critical situations that cannot be handled properly or timely by the INM algorithms; as a result, it can also generate exceptions that are sent to the operator when such violations occur.

3.4 INM Cross-Layer QoS

The CLQ (Cross-Layer QoS) is a management capability containing two software modules (see Figure 24). The main function of the CLQ capability is to provide performance indicators of specific links in the network by employing a set of measurements and to provide this information to other entities running in the network.

Comment [GN4]: It would be better to remove explainations about C/Java. Can't we just talk about measuring and publishing parts??

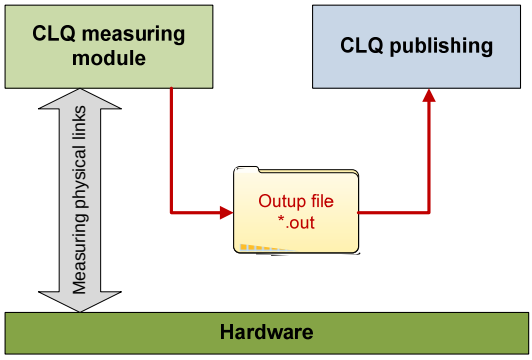


Figure 24 INM CLQ parts: measuring and publishing

One module is a specialized QoS software that performs measurements on the network links, above the MAC sub-layer, utilizing both active (to determine the OWD – One Way Delay) and passive (for determining the ATR – Available Transfer Rate) techniques. The results of the measurements are being stored in specific output files that are being read by the publishing CLQin order to update and make available the new values as local INM objectives. Once the measurements are being read a message is sent to all the subscriber entities running in the local or remote node.

As seen in Figure 25, any potential beneficiary, interested in the local objectives of a node (ATR or OWD) can use the publisher – subscriber paradigm in order to be constantly informed about the situation in the network. For doing so, the subscriber must get the handle of the CLQ module of a certain node, by specifying its IP address as an argument of the method `getInmElement()`. In the second step, it should use the function `subscribe()` with the proper arguments in order to initiate a subscription to the CLQ capability to be constantly informed whenever the objectives are updated. When the message “objective-changed” is received, the module can query the local objectives of the CLQ bundle by invoking the `getobjectives()` method. This method returns a vector containing the performance indicators of that node which is declared as containing InmObjective objects. The InmObjective class is offered by the INM framework.

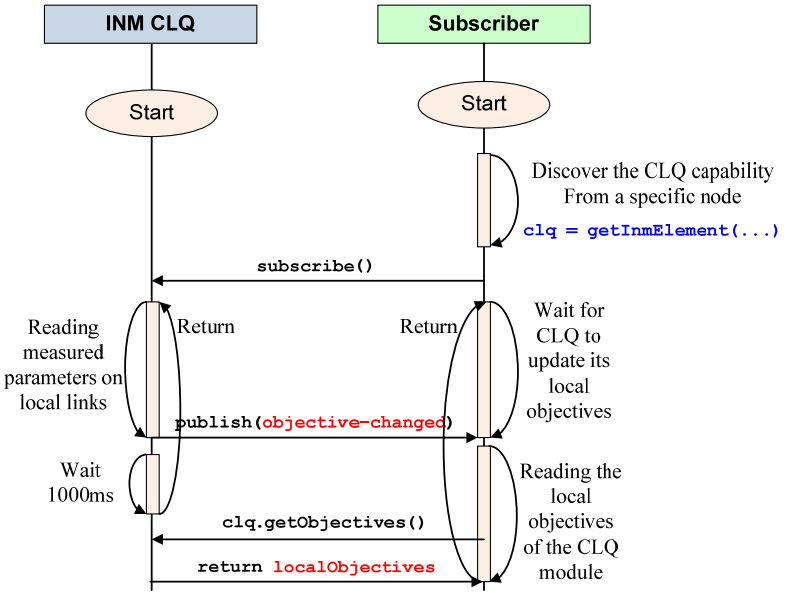


Figure 25 Message flows between INM CLQ and Subscriber

The possibility of performing measurements by request is another characteristic of the CLQ capability. Thus if any entity from a local or remote node requests a set of measurements at a specific time, it can invoke a method of the InmCrossLayer class.

Figure 26 shows the signalling messages needed by INM CLQ with respect to self-managed entities such as GP, NetInf or directly with hardware. The real-time INM Composite Metric (CM) is an immediate example of Cross-Layer QoS beneficiary. The preliminary formula used for an overall perspective of the links with the neighbours (for hop-by-hop data transport) was:

$$CM = \frac{k_0}{ATR[bps]} + \frac{OWD[s]}{k_1} + k_2 \times BER \qquad (1)$$

where ATR represents the Available Transfer Rate, OWD is One Way Delay, BER is Bit Error Rate, $k_0 = 10^9$ [bps], $k_1 = 10^{-5}$ [s] and $k_2 = 0$. This CM helps the management as criteria for triggering network-coding based GP activation, QoS-aware routing, etc. Regarding the demo "Cross-Layer QoS and Generic Path for Real-Time Congestion Control" presented in Section 2.3, a detailed description of the signalling between the nodes is given herein. Before activating the coding operations, some additional validations are being performed by the GP mechanism.

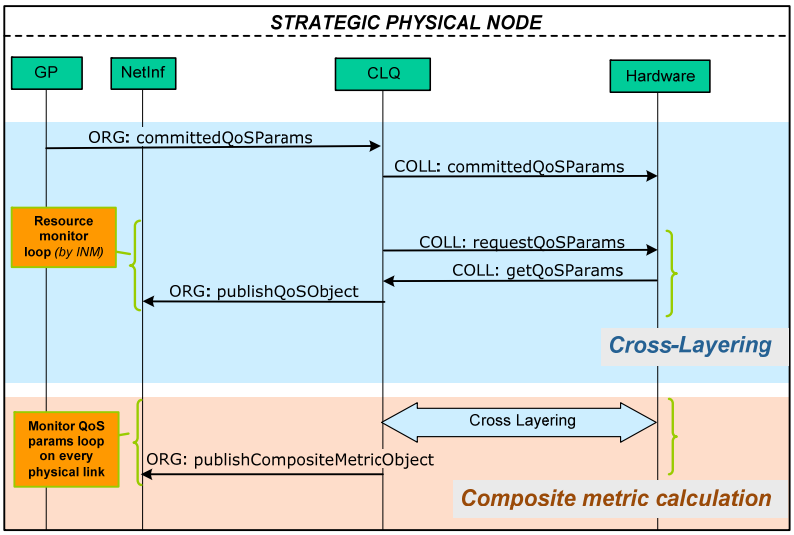


Figure 26 Messages exchanged between INM CLQ, GP, NetInf and the hardware

The conditions that need to be met by the network before starting the congestion avoidance operations are the following:

- **condition 1**: congestion on link R5-R6 need to be present – this information is provided by the statistics module (see Figure 27).
- **condition 2:** the available transfer rate on link R1-R3 must be higher than the data rate of the flow that travels on link R1-R5-R6-R4 (flow 1 from Figure 27):
  $ATR_{R1-R3} > D_{R1-R5-R6-R4}$
- **condition 3: t**he available transfer rate on link R2-R4 must be higher than the data rate of the flow that travels on link R2-R5-R6-R3 (flow 2 from Figure 27):
  $ATR_{R2-R4} > D_{R2-R5-R6-R3}$

Nodes R1, R2 are continuously checking conditions (2) and respectively (3) and when they are fulfilled, a signaling message is sent to R5. This node is the one that will be informed by the statistics module if there is congestion on link R5-R6 and if this happens, an activation flag will be sent to nodes R1 and R2, informing them to start the coding mechanism.

Node R5 is continuously checking congestion state and sends a signalling flag to R1 and R2 (Figure 28):

```
If  congestion = TRUE & Flag R1 = TRUE & Flag R2 = TRUE
      THEN Flag R5 = TRUE
      ELSE Flag R5 = FALSE
```

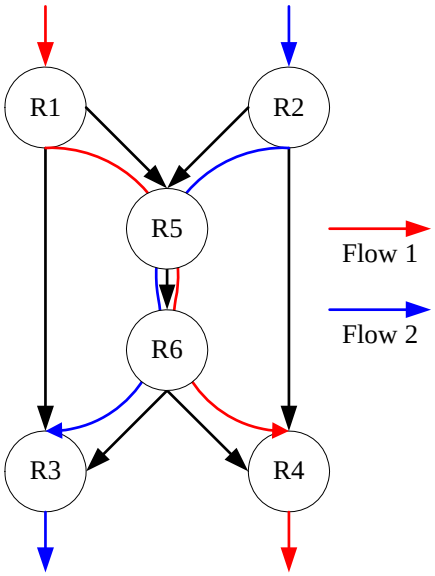


Figure 27 Butterfly topology testbed

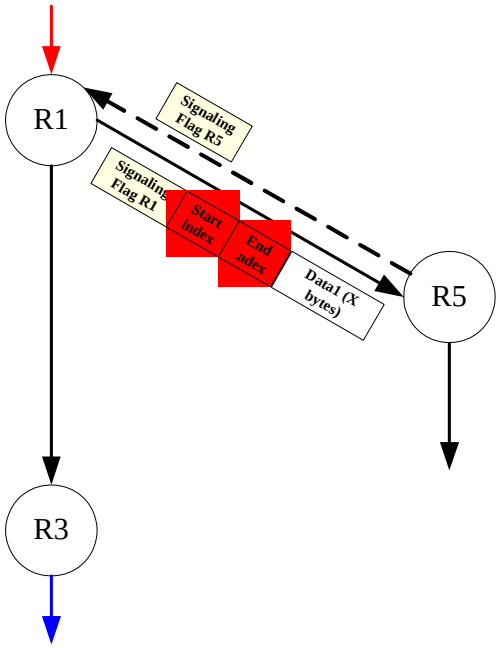


Figure 28 Signalling between encoding node R5 and the source node R1

The coded flow received by R6 from R5 contains signalling information that needs to be interpreted. If one set of Start – End indexes are zero, then R6 will implement forwarding only to one destination (means that coding is not activated) and if Start – End indexes are not zero, means that coding is activated in the network and coded packet will be multicast to both destinations. In nodes R3, R4 the packet header of the received flow includes the indexes of the bytes encoded by node R5. The decoding operations are being activated only if both Start and End indexes are not zero.

3.5 INM Network Wide Monitoring

The Generic Aggregation Protocol (GAP) is a protocol for continuous monitoring of network state variables (also called aggregates). Aggregates are computed from device counters using functions, such as SUM, AVERAGE and MAX. Examples of such variables are the total number of VoIP flows, and a histogram of the current load across routers in a network domain. GAP executes on an overlay that interconnects management processes on the devices. On this overlay, the protocol maintains a spanning tree and updates the network state variables through incremental aggregation.

We have implemented GAP in Java using three management capabilities. The “Overlay Agent” MC (labelled OA in Figure 29) creates and maintains an aggregation tree, the structure we use to compute the aggregate in a distributed manner, bottom-up. The “Aggregator Agent” MC (labelled AA in Figure 29) performs the computation of the aggregate. The “Variable Agent” MC (labelled VA in Figure 29) is responsible to capturing the local variables in a node that will be aggregated.

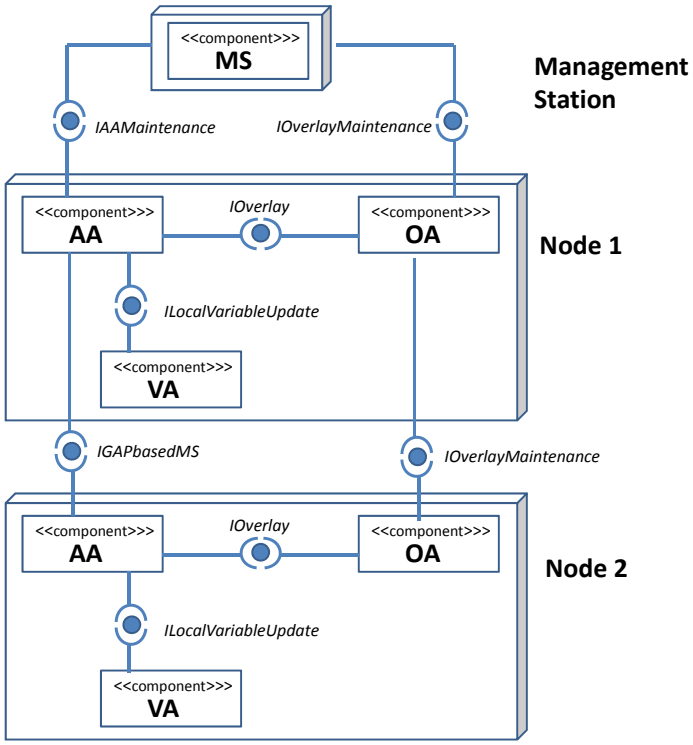


Figure 29 Implementation of GAP

Comment [GN5]: Text prints small.

Comment [GN6]: This prints dark in b&w.

When GAP runs, the first message exchanges are between instances of the “Overlay agent” MCs on different nodes. They exchange information through the collaboration interface in order to create and maintain the aggregation tree. The second group of message exchanges is among “Aggregator Agent” MCs on different nodes. They also use the collaboration interface to compute network state variables through incremental aggregation.

The consumer of the monitoring data specifies the aggregate through the organisation interface of the 'Aggregator Agent' MC. The data consumer must also specify through the organization interface the root of the aggregation tree. The aggregate will be available at that node.

GAP publishes, and therefore makes available to data consumers, an updates an aggregate using the function **void updateObjective(String Aggregate)** which is called by the root node whenever a new value for the aggregate is computed.

3.6 INM Anomaly detection

The distributed anomaly detection algorithm is a monitoring capability for local anomalies of links. This capability relies on an embedded test function that must be present in the network element by default and performs tests through a collaborative paradigm across adjacent nodes. Anomalies are meant as disruptive link failures or time deviations in the delay. To support this, the algorithm includes adaptation. This prototype is provided as a proof-of-concept of adaptation properties of distributed network management and the feasibility of a self-learning mechanism.

The anomaly detection consists of a public interface **AD** that is instantiated in each node of the network, as it implements a fully distributed algorithm (Figure 30). The **AD** interface

manages the data measurements performed for each connection in the nodes. Further, it manages message exchanges between nodes, such as algorithm control messages and probe requests. The connections are modelled individually via the **ADModule**, which keeps a statistical model **ADModel** for the certain connection towards a neighbouring node. The return values returned by the **AD** interface relate to outgoing messages and time intervals for various algorithm events. Note that the implementor of the network function needs to provide means for measuring probe response delays, as this functionality is not included in the anomaly detection prototype.

In runtime, the **AD** relies mainly on two callback functions that return the time to the next event and one or more outgoing messages related to a certain neighbouring node (Figure 30). First, scheduled events, such as probe requests and other events, are managed by the use of **scheduledCallback(…)**. Second, **incomingMsgCallback(…)** is used for managing algorithm control messages. The function manages probe delay inputs from performed measurements, and control messages exchanged between collaborating nodes during localisation of faults. The time returned is used to schedule the next call to **scheduledCallback(…)**. Depending on the algorithm state, both functions return time-out intervals and probing intervals based on the statistical model related to a certain connection, measured from the node to one of its neighbours.

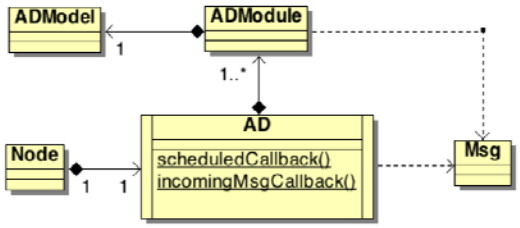


Figure 30 Implementation of distributed anomaly detection for one node

An example of the interaction between three nodes using AD-instances is shown in Figure 31 below. Here node 1 is unable to communicate with node 3; the anomaly was discovered after node 1 sent a series of unsuccessful probes to the neighbouring node. In order to verify the status of node 3, the detecting node 1 requests a probe test on node 3 performed by node 2. Node 2 immediately sends a first probe to node 3, which responds immediately upon reception of the probe. Node 2 receives the response from node 3, and immediately returns a control message back to node 1 in order to inform about the outcome of the requested probe test.

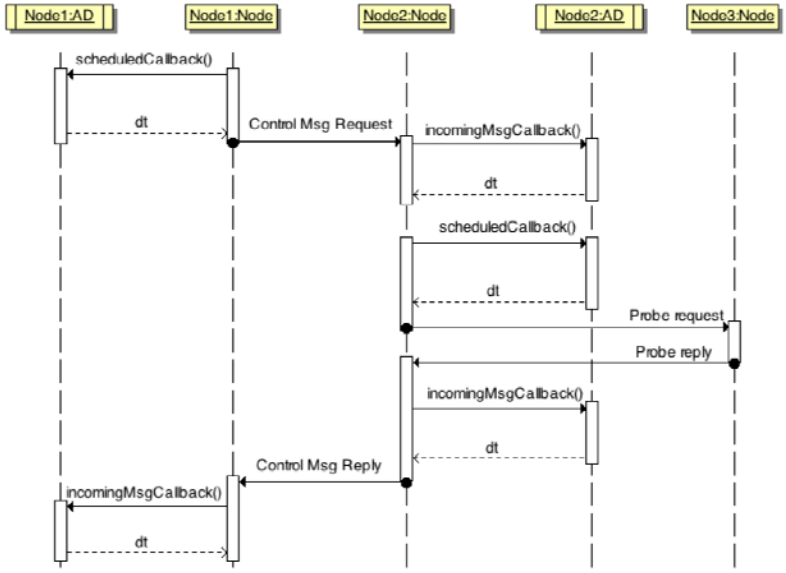


Figure 31 Interaction between nodes implementing anomaly detection

The anomaly detection module provides a good example of coupling a management function (in this case a probe) with the network function. The probe used by the anomaly detection in fact relies on external method provided by the network function: this can be a simple method like an ICMP request on IP networks or a loopback message on a virtualized network. From an architectural point of view, the anomaly detection module is integrated as an MC, exposing detected faults and anomalies to the network via the collaborative interface, whereas high-level requirements are set via the organizational interface.

3.7 INM Congestion Control

The INM Congestion Control (INM CC) is an approach that brings congestion control mechanisms inside the network and does not try to solve congestion problems in an end-to-end manner, e.g. as done in TCP. Based on pulse-coupled oscillators each router maps its queue filling level to a frequency and distributes this frequency along the routing path. The pulse-coupled oscillators will then synchronize themselves according to the highest frequency within this chain, which in turn reflects the congestion of the available path. Based on this congestion information the sending node can then easily select the most appropriate path to forward its data.

The implementation of the distributed congestion control algorithm is composed of two internal modules. **Error! Reference source not found.** visualizes these two modules, the most important functions that are needed and their relation to other management capabilities.

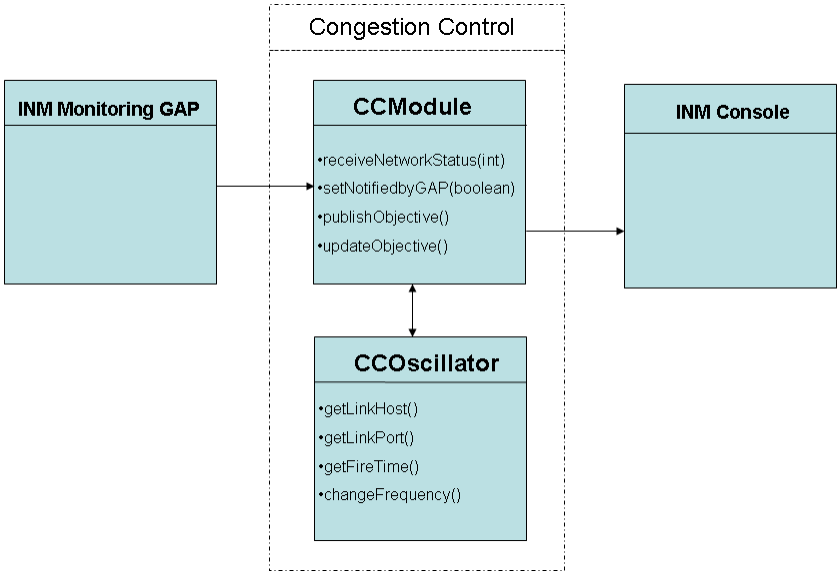


Figure 32 Implementation of Congestion Control Module

The first module is called *CCOscillator* and its basic functionality can be easily described by its most important functions. Tuple *getLinkHost()* and *getLinkPort()* represent the address of the each queue on a router (incoming and outgoing links). This information will be used to distribute the queue filling level along the path. Functions *getFireTime()* and *changeFrequency()* are important for the synchronization behaviour of the developed approach. They will make sure that the distributed frequency along a path will be the same for each interface and continuously adapted according to the highest frequency. Again, it is important to notice that the frequency is just a mapping of the queue filling level, which in turn represents the overall congestion for a certain routing path.

The second module is called CCModule, which implements the load balancing logic and interconnects the module to the INM Core (*publishObjective()*). To avoid wasting any resources for exchanging network congestion information if the network itself is not congested at all, INM CC will be deactivated by default. If the network starts getting congested and an overall network congestion threshold computed by GAP is crossed (e.g. 65%), GAP will notify INM CC to start load balancing by calling function *setNotifiedbyGAP(boolean)*. The CCModule will then start distributing the outgoing traffic based on the congestion level of its available paths computed by receiveNetworkStatus(int). Finally, the CCModule will then continuously update the INM Console about the congestion level by calling *updateObjective()*.

4 Conclusions

This deliverable described the INM integrated prototype developed inside WP4. The prototype served two purposes. The first one is to provide a proof of concept of selected INM functions; the second one is to demonstrate the benefits of INM in potential exploitation scenarios.

The integration of the INM functions followed a scenario-driven approach. We introduced a reference scenario, where INM is presented as enabler for support in operating networks, support advanced services like emergency or multimedia real-time. We then proposed three specific scenarios that apply INM to other technologies in 4WARD. One is WP2/4 scenario that shows interoperability issues in emergency scenarios. The second is one real-time

monitoring of QoS for adaptation of Generic Path properties. The third one is optimization of virtual resources. These three scenarios serve as showcase for the benefits of distributed architecture, real-time monitoring and objective-driven management.

Based on the results of the previous deliverable D-4.3, this document shows first the definitions of the main interfaces of the INM entities: they are used to enforce objectives and to communicate failures between the operator and the distributed architecture. Second a list of INM functions is described and a detailed description of internal mechanisms is described in form of modules or interaction diagrams.

The experience reported in this document shows how the INM approach can be transferred to a real implementation. This document has considered a selected set of INM functions as concrete examples and proposed the INM runtime as starting framework for future exploitation of INM.

References

[1] "D-4.3 In-network Management Design", Alberto Gonzalez (editor) et al., FP7-ICT-2007-1-216041-4WARD, 26 May 2010, Revision 2.0, http://www.4ward-project.eu/.

[2] "D-4.5 Evaluation of the in-network management approach", Alberto Gonzalez (editor) et al., FP7-ICT-2007-1-216041-4WARD.

[3] "D-2.3.0 Architectural Framework: new release and first evaluation results", María Ángeles Callejo, Martina Zitterbart (editors) et al., FP7-ICT-2007-1-216041-4WARD, 14 January 2010.

[4] "D-5.2.0 Description of Generic Path Mechanism", Th.Biermann (editor) et al., FP7-ICT-2007-1-216041-4WARD, 5 May 2009, Revision 2.0, http://www.4ward-project.eu/.

[5] "Cross-Layer QoS and Its Application in Congestion Control", A.B.Rus, M.Barabas, G.Boanea, Z.Kiss, Z.Polgar & V.Dobrota, 17th IEEE Workshop on Local and Metropolitan Area Networks LANMAN 2010, Long Branch, NJ, USA, May 5-7, 2010, http://www.ieee-lanman.org/

[6] "Preliminary Implementation of Point-to-Multi-Point Multicast Transmission Based on Cross-Layer QoS and Network Coding", Z.Polgar, Z.Kiss, A.B.Rus, G.Boanea, M.Barabas & V.Dobrota, 17th International Conference on Software, Telecommunications & Computer Networks IEEE SOFTCOM 2009, Split-Hvar-Korkula, Croatia, September 24–26, 2009, pp.131-135

[7] "D4.1 Management & Knowledge Planes Initial Design", Lefteris Mamatas (editor) et al., FP7-ICT-2007-1-216404 Autonomic Internet (AutoI) Project, Revision 2.0