Objective FP7-ICT-2007-1-216041/D-4.3

The Network of the Future

Project 216041

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

D-4.3
In-network management design

Date of preparation: **10-05-26**
Start date of Project: **08-01-01**
Project Coordinator: **Henrik Abramowicz**
Ericsson AB

Revision: **2.0**
Duration: **10-06-30**

Document Properties[1]:

Document Number[2]:	FP7-ICT-2007-1-216041-4WARD / D-4.3
Document Title:	**In-network Management Design**
Document responsible:	Alberto Gonzalez Prieto, KTH.
Author(s)/editor(s):	Gerhard Hasslinger (DT), Catalin Meirosu, Srdjan Krco, Sidath Handurukande (EAB), Fabian Wolff, Thomas Hirsch, Jens Tiemann, Michael Kleis (Fraunhofer), Susana Sargento, Lucas Guardalben (IT), Karl Palmskog, Fetahi Wuhib, Mads Dam, Rolf Stadler (KTH), Dominique Dudkowski, Giorgio Nunzi, Bioniko Tauhid, Marcus Brunner (NEC), Changpeng Fan, Henning Sanneck (NSND), Vitor Mirones (PTIN), Rebecca Steinert, Daniel Gillblad (SICS), Leonard Pitu (SROM), Avi Miron (Technion), Sławomir Kukliński (TPSA), Christopher Foley (TSSG), Virgil Dobrota, Bogdan Rus, Emanuel Puschita, Tudor Palade (TUCN), Manolis Sifalakis (UniBasel).
Target Dissemination Level[3]:	PU
Status of the Document:	Final (Outline - Draft – Final)
Version	2.0

Revision History:

Revision	Date	Issued by	Description
v.2.0	2010-05-26	Alberto Gonzalez Prieto	Revised version.
v.1.0	2009-01-15	Alberto Gonzalez Prieto	First public version.

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

[1] Input of Title, Date,Version, Target disseminationlevel, 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

Abstract:

This deliverable reports on our work developing the novel paradigm of In-network Management (INM), aimed specifically at the effective management of large, dynamic networks. Its basic enabling concepts are decentralization, self-organization, and autonomy. The idea is that management tasks are delegated from management stations outside the network to the network itself. The INM approach therefore involves embedding management intelligence in the network, or, in other words, making the network more intelligent. The managed system now executes management functions on its own. It performs, for instance, reconfiguration or self-healing in an autonomic manner.

This deliverable describes the main results of WP4 achieved during the second year of the project and it complements deliverable D-4.1, which presents use cases illustrating the potential of INM capabilities. First, it contains the design of the INM framework, an enabler of management functions that defines a set of architectural elements from which any distributed and embedded management structure can be created. Second, it presents algorithms for situation awareness in real-time and self-adaptation that provide a subset of management functions that we regard as important and challenging for the management of the future Internet. Third, it outlines our work towards an INM prototype due in June 2010.

Finally, we report the results of our close collaboration with other WPs in the 4WARD project. Specifically, we describe the support INM offers to VNets (WP3), GPs (WP5) and NetInf (WP6). We also present how INM can benefit from NetInf. The integration of the INM framework into the 4WARD architecture (WP2) is discussed too.

Keywords:

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

0 Table of Contents

1 Executive Summary

Work Package 4 (WP4) works towards the definition of novel management instruments to operate the future Internet. The approach we propose is called In-Network Management (INM). Its basic enabling concepts are decentralization, self-organization, and autonomy. The idea is that management tasks are delegated from management stations outside the network to the network itself. The INM approach therefore involves embedding management intelligence in the network, or, in other words, making the network more intelligent. The managed system now executes management functions on its own. It performs, for instance, reconfiguration or self-healing in an autonomic manner.

One can consider a network management system as executing a closed-loop control cycle, whereby the network state is estimated on a continuous basis (i.e., situation awareness), and, based on this estimation, a process dynamically determines a set of actions that are executed on the network in order to achieve operational objectives (i.e., adaptation).

The work we have developed contributes to both parts of the control cycle. Clearly, we can not cover completely all aspects of network management. Therefore, we have selected a set of management functions that we regard as the most important and challenging for the management of the future Internet. Specifically, we have developed algorithms for situation awareness that address real-time monitoring of network-wide metrics, group size estimation, topology discovery, data search and anomaly detection. We have also developed algorithms for self-adaptation, giving special attention to self-adaptation in the context of other areas of research in 4WARD, such as VNets (WP3) and GPs (WP5), WP6. Furthermore, we have developed self-adaptive solutions for the self-organization of the management plane, aiding configuration planning. Finally, also in the context of self-adaption, we have developed a framework for designing self-adapting network protocols inspired in chemical processes

In order to support INM algorithms, we have created an INM framework. The framework is an enabler of management functions and it defines a set of architectural elements from which any distributed and embedded management structure can be created.

We also report on our on-going work towards an INM prototype. It will serve as a proof-of-concept for the INM approach, the framework and selected algorithms. The prototype will also serve as a basis for a demonstrator.

Our WP has worked in close collaboration with other WPs in the 4WARD project. We have identified synergies between INM and other areas of research in the project. Specifically, we have investigated the support INM offers to VNets and GPs. We have also investigated how INM and NetInf (WP6) can benefit from each other offering a range of functionalities to each other. Finally, we have integrated the INM framework into the 4WARD architecture (WP2).

2 Introduction

Work Package 4 (WP4) is working towards the definition of novel management instruments to operate the future Internet. It defines (i) a new architecture to support distributed management operations and (ii) a set of distributed management algorithms.

2.1 Motivation of this Document

Today's centralized network management will no longer be applicable to the large-scale networks and services foreseen in the future. This was shown through use cases in our previous deliverable D4.1. New approaches for management are needed.

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

This deliverable summarizes the solutions developed by WP4 during the second year of the project. ([D4.2] presented the solutions designed during the first year). A selected set of these solutions will be further evaluated and implemented through a prototype in the remainder of the project.

2.2 Objective of this Document

The objective of this deliverable is to describe the work done by WP4 in the second year of the project. Specifically, it describes:

- the **INM framework**. This framework enables the deployment and execution of distributed management algorithms,
- the **distributed management algorithms** developed during the last 12 months,
- the results of the **collaboration with other WPs** within 4WARD,
- the on-going work on the **INM prototype.**

2.3 Structure of this Document

Chapter 3 provides an overview of the work done during the first two years and presents the key results and achievements of WP4. This chapter is intended as a stand-alone text that provides the big picture of the contents of the deliverable. Chapter 4 summarizes the motivation for INM and the scope of the work in WP4. Chapter 5 presents the main concepts of the INM framework. Chapter 6 presents algorithmic solutions for real-time situation awareness. Chapter 7 presents adaptation schemes. Chapter 8 outlines on-going work towards an INM prototype and demonstrator. Chapter 9 describes our work in collaboration with other WPs within 4WARD. Chapter 10 compares our work to that done in other projects with a related focus. Finally, Chapter 11 concludes the deliverable.

3 Overview of the Deliverable

This chapter summarizes the work and main results of WP4 achieved during the second year and reported in this deliverable. The chapter can be read as a stand-alone text and is intended to provide in a concise manner the big picture and key aspects of our work.

3.1 INM for the Future Internet

Work Package 4 (WP4) is working towards the definition of novel management instruments to operate the future Internet. It defined (i) a new framework to support distributed management operations and (ii) a set of distributed management algorithms, INM algorithms.

3.1.1 Limitations of Existing Approaches

Traditional management solutions typically execute in management stations that are outside the network. These stations interact with the managed devices, mostly on per-device basis, in order to execute management tasks. During network operation, for instance, a management station periodically polls individual devices in its domain for the values of local variables, such as devices counters or performance parameters. These variables are then analyzed on the management station by management applications to determine the set of actions to execute on each on the managed devices.

This paradigm of interaction between the management system and the managed system has been used by traditional management frameworks and protocols, including SNMP, TMN and OSI-SM. This paradigm has proved fairly successful for networks of moderate size, whose states evolve slowly and thus their configuration rarely needs to be changed and no rapid interventions are required. Many of today's emerging networks depart from these assumptions. We envision that in the future Internet, particularly with its wireless and mobile extensions, networks will include millions of network elements, whose state will be highly dynamic and whose configuration will need to adapt on a continuous basis.

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

3.1.2 The INM Approach

Today's deployed management solutions have two main characteristics. First, managed devices generally present simple and low-level interfaces. Second, management stations interact directly with managed devices typically on a per-device basis. There is no interaction among managed devices for management purposes and those devices have little autonomy in making management decisions. As a consequence, managed devices are "dumb" from a management standpoint.

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

In 4WARD, we propose a novel approach to network management. By this we mean a way of envisioning and engineering management concepts and capabilities that abandons the two characteristics described above.

The approach we propose in the 4WARD project is called In-Network Management (INM). Its basic enabling concepts are decentralization, self-organization, and autonomy. The idea is that management tasks are delegated from management stations outside the network to the network itself. The INM approach therefore involves embedding management intelligence in the network, or, in other words, making the network more intelligent. The

managed system now executes management functions on its own. It performs, for instance, reconfiguration or self-healing in an autonomic manner.

In order to realize this vision, a management entity with processing and communication functions is associated with each network element or device, which, in addition to monitoring and configuring local parameters, communicates with peer entities in its proximity. The collection of these entities creates a management plane, a thin layer of management functionality inside the network that performs monitoring and control tasks.

The potential benefits of the INM paradigm include the following properties:

- A **high level of scalability** of management systems. For instance, in terms of short execution times and low traffic overhead in large-scale systems. This will allow for effective management of large networks.
- **Fast reaction times** in response to faults, configuration changes, load changes, etc. This increases the adaptability of the network. Together with embedded functions, this will lead to with a high level of robustness of the managed system.
- A **high business value** for INM technologies. The higher degree of automation of INM solutions, compared to traditional ones, permits the reduction of operational expenditures. Capital expenditures are also reduced since INM technologies enable a more efficient use of the network infrastructure.

A possible drawback of this paradigm is that processing resources for management purposes must be available in the network elements (or in the managed system), which will potentially increase cost and network element complexity.

3.1.3 Scope of the Work in 4WARD

One can consider a network management system as executing a closed-loop control cycle, whereby the network state is estimated on a continuous basis (i.e., situation awareness), and, based on this estimation, a process dynamically determines a set of actions that are executed on the network in order to achieve operational objectives (i.e., adaptation).

The work developed in 4WARD contributes to both parts of the control cycle. Specifically, we have developed algorithms for situation awareness that address real-time monitoring of network-wide metrics, group size estimation, topology discovery, data search and anomaly detection. We have also developed algorithms for self-adaptation, giving special attention to self-adaptation in the context of other areas of research in 4WARD, such as VNets (WP3) and GPs (WP5). Furthermore, we have developed self-adaptive solutions for the self-organization of the management plane, aiding configuration planning. Finally, also in the context of self-adaption, we have developed a framework for designing self-adapting network protocols inspired by models of chemical processes.

In order to support INM algorithms, we have created an INM framework. The framework is an enabler of management functions and it defines a set of architectural elements from which any distributed and embedded management structure can be created. Currently, we are implementing an INM prototype and demonstrator. It will serve as a proof-of-concept for the INM approach, the framework and selected algorithms.

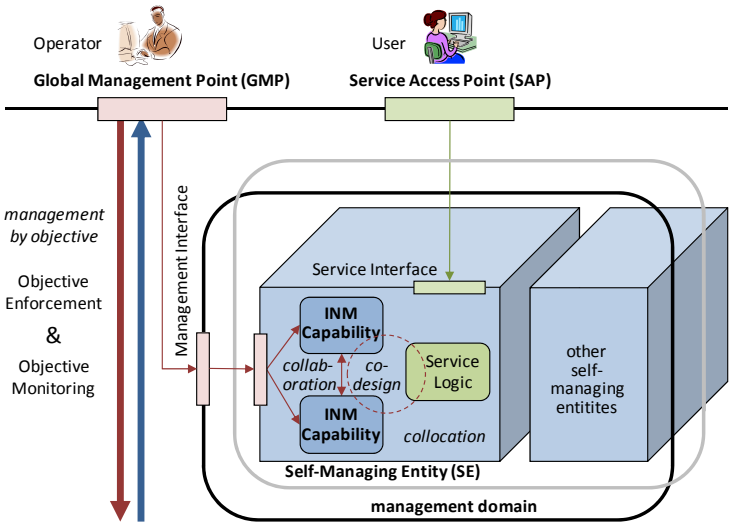
Our WP has worked in close collaboration with other WPs in the 4WARD project. We have identified synergies between INM and other areas of research in the project. Specifically, we have investigated the support INM offers to VNets and GPs. We have also investigated how INM and NetInf (WP6) can benefit from each other offering a range of functionalities to each other. Finally, we have integrated the INM framework into the 4WARD architecture (WP2).

3.2 INM Framework

The architectural framework for INM [Dud09c] (i.e., the INM framework) supports INM operations in the future Internet by the means of a highly distributed architecture. The main

objective is the design of INM functions that are located close to the management services, in most of the cases co-located on the same nodes. The framework proposes the fundamental principles and constructs that state how to design and operate networks according to the INM paradigm.

The figure below illustrates the developed INM framework's main concepts. On the operator side, the *global management point* (GMP) is the high-level entry point into the management of a virtual network. The GMP is the only management interface visible to the operator and provides the highest level of abstraction by means of objectives.



Overview of the INM Framework

In a first hierarchical refinement, the global management point provides access to one or more *management domains*, each allowing access to a well-defined subset of the embedded management plane. An INM domain is also the main abstraction that enforces security by restricting access to only those INM functions that an operator is allowed to handle. Multiple domains may exist at any time during network operation, their configuration may change and they may be set up and torn down dynamically over time.

While the purpose of management domains is to extract a well-defined management subject in terms of structural and functional characteristics, *self-managing entities* (SEs) are service-oriented and encapsulate self-management functions of individual services. SEs provide the means for embedding a set of generic properties and abstract interfaces that enable network operation with only high-level intervention from the operator. For that, SEs collaborate with one another and enforce objectives on the service level in order to meet the service-specific objectives dictated by the operator's high-level objectives.

Each self-managing entity contains multiple management capabilities (MCs), which implement the actual self-adaptive INM algorithms on a fine granularity. All INM algorithms are implemented at this level, and run as a set of MCs. While containment of MCs within SEs is one possible realization, management capabilities may also exist stand-alone. This configuration is useful for MCs that encapsulate more generic management algorithms, for example, aggregation algorithms for monitoring discussed in later sections.

The enforcement and monitoring of objectives occurs along the hierarchy of the previously described elements. Each objective is specified on an abstract level by the operator at the level of the global management point and split into sub-objectives via management domains and self-managing entities until reaching individual management capabilities. Correspondingly, objectives and performance are monitored and aggregated towards the global management point in the opposite sequence of elements.

3.3 INM Real-time Situation Awareness

Situation awareness consists in estimating the state of the network system. This is a very wide field and, clearly, we can not cover it completely. We have worked towards providing a selected a set of functions that we regard as important and challenging for the management of the future Internet. The result is a set of complementary distributed algorithms that provide views of the network state in real-time. This functionality includes real-time monitoring of network-wide metrics, group size estimation, topology discovery, data search, and anomaly detection. These algorithms provide the necessary input to the self-adaptation mechanisms.

Regarding the **monitoring of network-wide metrics**, an important task is to detect a threshold crossing of a metric, for instance to determine that some network-wide metric, such as average or peak load, has been crossed, indicating a problem that may need attention. We have developed solutions for threshold detection based on both tree-based and gossip-based underlying protocols [Wuh08] [Wuh09a] [Wuh09b]. The overhead of our solutions is minimal when the aggregate values are far from the thresholds. This is a very attractive feature for many applications where the thresholds indicate exceptional network conditions that are not expected to frequently arise, as this means that, under normal operating conditions, overhead from threshold detection can be neglected.

For providing **group size estimation**, we have engineered NATO! [Coh09], a statistical probability scheme for estimating the size of a group of nodes affected by the same event without explicit notification from each node, thereby avoiding feedback implosion. An efficient solution for this task permits, for instance, monitoring the operating conditions of a large-scale network by computing the share of nodes whose performance is above (or below) a given threshold. The main idea in our solution is that after the event takes place, every affected node waits a random amount of time taken from a predefined distribution, before sending a report message (RPRT). Simulation results show that with only 20 feedback messages (coming from a group of 100 or 10,000 affected nodes), the estimation error is about 5 percent, and after error correction, the error is eliminated.

We have engineered “Hide and Seek” (H&S) [Gua09], a novel algorithm for **topology discovery**. In highly dynamic scenarios, like the ones we target, the need for efficient topology discovery is particularly important. There are two roles in H&S: seeker and hider. A seeker node sends directional contact messages to its neighbourhood using a probabilistic eyesight direction (PED) function. This function aims at narrowing the directions through which contact messages are sent. Once contacted, hider and seeker synchronize their knowledge, keeping track of each other. The hider becomes a new seeker and the process is repeated until all nodes have been contacted. Our evaluation shows that the message overhead of H&S is lower than that of flooding-based algorithms.

In the context of **data search**, we have investigated the efficiency of random walks and flooding for exploring networks, based on case studies evaluated by simulation and transient analysis [Hass08] [Hass09]. Specifically, we have considered, single random walks, multiple random walks, biased random walks and flooding.

We have developed a distributed approach to adaptive **anomaly detection** and collaborative fault-localisation [Ste09a] [Ste09b]. Clearly, the relevance of this management task is higher in the dynamic future Internet than in traditional, more stable, scenarios. The statistical method used is based on parameter estimation of Gamma distributions obtained by measuring probe response delays. The idea is to learn the expected probe response delay and packet drop rate of each connection in each node, and use it for parameter adaptation such that the manual configuration effort is minimised. Our simulation results show that we achieve robust anomaly detection with few false positives.

We have also investigated **algorithms aspects** in the context of distributed network monitoring. First, we have evaluated different algorithmic approaches to distributed aggregation [Wuh08] [Wuh09a] [Wuh09b]. Specifically, we have focused on two main classes of algorithms, tree-based and gossip-based algorithms. Our theoretical and experimental

investigations indicate that in a number of respects tree-based protocols are superior to gossip-based aggregation protocols. For instance, we consistently found that the overhead a tree-based solution is about an order of magnitude lower than that of a gossip-based one. In addition, we found that the problem of making gossip protocols robust is far more challenging than in the case of tree-based ones. Our conclusion is that, at this time, tree-based approaches are preferable to gossip-based ones for INM applications.

Second, we have developed and evaluated stochastic models of tree-based aggregation under churn, i.e., where network nodes may be dynamically removed from, or join, the network. We have developed several models. The models estimate network properties at varying levels of detail such as the number of nodes at a given tree level, or the aggregate held by a node at a given level in expectation. The models have been validated by simulation for varying churn rates and protocol rates.

Finally, we have developed **secure versions of our algorithms** for monitoring of network-wide metrics [Kre09]. These versions are able to execute without providers private information leaking to outsiders. This is particularly important for network management information, as this generally contains lots of information about the configuration, operation, load, and performance, of the providers' internal network. We have, so far, examined the case of passive, “honest-but-curious” attackers. We provide solutions for the standard aggregation functions sum, average, min, and max. For sum and average, we give a protocol that is information theoretically secure against a passive adversary, and which requires only one additional round compared to non-private protocols for computing sums. For min and max, we present two solutions. The first is computationally secure, and the second is information-theoretically secure.

3.4 INM Self-adaptation

Self-adaptation is a key feature for INM. Following the INM vision, it is the network itself that takes proactive or reactive actions in order to adapt to changing networking conditions and to continuously meet the network operator objectives.

The focus of our work has been the investigation of selected aspects of self-adaptation, including self-adaptive solutions for VNets and GPs, the self-organization of the management plane, and configuration planning. We have also developed a framework for designing self-adapting network protocols. In addition, we have investigated the characteristics of self-adaptation in the context of INM capabilities, a key architectural element of the INM framework.

INM self-adaptation provides **support to other research groups in 4WARD**, such as VNets and GPs. In order to support Vnets, we have developed INM algorithms that adapt resource requests to the network capabilities. We have identified functionalities INM can provide to GP and have developed solutions. For instance, we have designed a **congestion control** scheme which exhibits emergent behaviour. The concept is adapted from a completely different domain of pulse-coupled oscillators, which synchronize its phase emergently. The scheme does not rely on communication between end systems and routers in order to identify and remedy a congestion situation. Another line of work that supports GPs is one where we investigated **adaptive routing in wireless multi-hop networks**. In our solution, routing protocols and parameters are dynamically selected based on networking conditions.

In the context of the **adaptive self-organization of the management plane**, we have proposed an abstraction layer for aggregating events [Min09]. The degree of aggregation performed is dynamically adjusted, based on the operator objectives that can be tuned at run-time.

A key aspect of self-adapting algorithms is that they must guarantee that **configuration changes** lead to the desired network state and behaviour. With this objective in

mind, we have investigated a mechanism for predicting the outcome of a configuration change.

We have proposed a methodology based on chemical networking for **designing INM self-adapting algorithms**. The goal is to create robust protocol implementations which can continuously adapt to the network dynamics and enable resilient operation.

Finally, we have investigated the characteristics of self-adaptation in the context of INM capabilities. Specifically, we have investigated how MCs are internally organized in order to facilitate self-adaptation.

There are three possible design options for the self-adaptation control loop. First, fully embedding the self-adaptation control loop within the MC in a monolithic manner. Second, designing the self-adaptation control loop so that it is an identifiable entity inside the MC. Third, dedicating an MC for implementing the self-adaptation control loop, which enables self-adaptive behaviour of other INM MCs that are otherwise not self-adaptive.

We have developed a set of guidelines for selecting the most appropriate self-adaptation design option. The key findings are that the two first design options are a better fit for a fully distributed INM architecture, whereas the third design option is more appropriate for centralized approaches (or at least less distributed ones). Particularly, for very large networks, a distributed INM architecture is the only scalable option, and thus, the first two design options are preferred. In addition, we have identified that when an INM algorithm is complex and requires significant amount of processing/storage resources, it might be more appropriate to locate it in a designated SE that is capable of hosting it, thereby preferring the third design option.

3.5 Integration between INM and other WPs in 4WARD

The work on INM has contributed to several other areas of work performed within 4WARD. Here, we will briefly discuss how the INM framework has affected the general 4WARD architecture, how it can make use of networking of information, INMs role in virtual networks, and how INM algorithms can contribute to the management of GPs.

4WARD is exploring the development of a design process for combining existing or specifying new networks with customized architectures. INM fits into this design process within the *Governance* and *Knowledge strata*. The Governance stratum is related to a management domain and the Self Managing Entities that exist within it, and specifies the domains management objectives. The information produced by Management Capabilities is fed into the Knowledge stratum, which processes the information, presents it to operators, and feeds it back to the Governance stratum. Further, the INM algorithms we have developed serve as design patterns for solving different monitoring and adaptation problems. These algorithms can be key components of a design patterns repository and be included at the network design stage.

Network management functionality produces a significant amount of information, through counters, events/alarms and configuration parameters that determine system state during operation. 4WARD proposes a generic information-centric approach to networking, Networking of Information, where the identity of an information object is used as a reference to retrieve it from any node that might store the full information object. As INM processes are implemented in a distributed manner and need efficient access to information and information storage capabilities, NetInf is a suitable candidate for management information dissemination, while INM can also serve as an important use-case for networking of information. Different types of management information required by processes within the INM framework were identified in [Bru08].

In the context of virtual networking, INM concepts apply in two places. First, INM provides management operations to the base infrastructure. Second, INM capabilities and functions are implemented in accordance with the network architecture running within a virtual

network. Management capabilities in both places might need to interact, e.g. for the purpose of optimization or locating a network disturbance. INM addresses several issues within a virtual networking setting. The decentralized operation of the infrastructure achieves a well-balanced resource usage and concludes possible virtual node re-location. Further, the virtual links are subject to resource management actions in order to guarantee the bandwidth and QoS requirements [D4.2]. Finally, the issues involved in root-cause analysis and fault management are of major importance in a carrier-grade virtual networking environment.

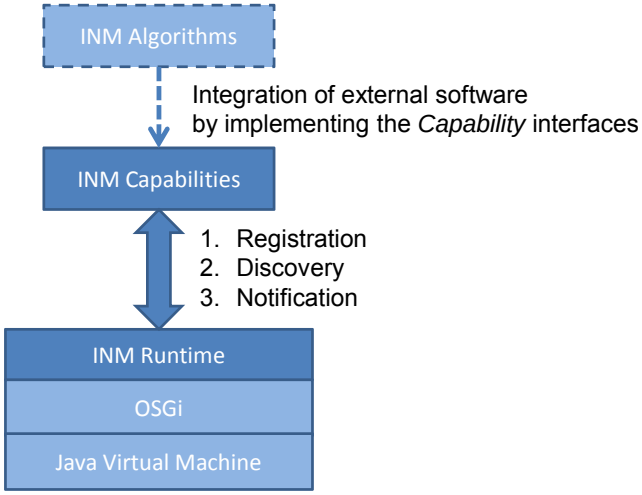
For Generic Paths, INM provides two key mechanisms: distributed real-time monitoring and resource adaptation. These mechanisms can be used for multiple purposes in path management, such as network state evaluation, distributed resource management, decision making for routing strategies, anomaly or fault detection, and fast failure recovery. INM also contributes to generic path routing strategies by providing information on the quality of routing paths, e.g. through providing accurate link availability statistics and through collaborative measurements, which differ significantly from simple path estimation approaches incorporated in present routing protocols. The generic path mechanism would collaborate with INM entities to provide support for real-time traffic path and resource availability monitoring, also across composite generic paths.

3.6 INM Prototype

We are currently implementing an INM integrated prototype and demonstrator. It will serve as a proof-of-concept for the INM approach, the framework and selected algorithms.

The prototype includes a console, which is the instrument given to operators for management. The console visualizes the actual values of key performance indicators and makes it possible to change the operational objectives for the network. The console typically shows indicators and objectives that are aggregated from different sources, rather than reporting a per-node view of the network.

Our implementation is depicted in the figure below.



Mapping of Algorithm Design into INM Platform.

The prototype is written in Java and built on top of the OSGi framework. The INM Runtime library provides a set of basic functions to INM capabilities (i.e., implementations of INM algorithms). These functions permit INM capabilities to register and discover other capabilities.

Communication between capabilities can be performed in two ways: using remote method invocation and in a publish/subscriber way.

3.7 Achievements during the Second Year and Conclusions

The main achievements and results of WP4 (during the second year of the project) towards the management of the future Internet can be summarized as follows.

INM Framework

- We have finalized the INM framework, which is an enabler of INM functions. Three main elements, namely, management capabilities, self-managing entities, and management domains were defined, from which complex management functions can be constructed.

INM Algorithms

- We have developed a set of INM algorithms for situation awareness that address real-time monitoring of network-wide metrics, group size estimation, topology discovery, data search and anomaly detection.
- We have also investigated algorithms aspects in the context of distributed network monitoring. Specifically, we have evaluated different algorithmic approaches to distributed aggregation and we have developed and evaluated stochastic models of tree-based aggregation under churn.
- We have developed secure versions of our algorithms for monitoring of network-wide metrics.
- We have developed a set of INM algorithms for self-adaptation, giving special attention to self-adaptation mechanisms required in the context of other areas of research within 4WARD, such as VNets (WP3) and GPs (WP5).
- We have developed self-adaptive solutions for the self-organization of the management plane and for aiding configuration planning.
- We have developed a framework for designing self-adapting network protocols inspired in models for chemical processes.

INM Prototype

- Currently, we are implementing an INM prototype. It will serve as a proof-of-concept for the INM approach, the framework and selected algorithms. The prototype will also serve as a basis for a demonstrator.

Dissemination

Based on results from work in this WP since the start of the project, 33 papers have been published in international conferences, journals and magazines, such as IEEE Transactions on Network and Service Management, IFIP/IEEE International Symposium on Integrated Network Management, and IEEE ICC. In addition, 11 are currently under review.

Furthermore, INM work is highly visible in the network management research community. Results are disseminated in the key conferences and events presenting technical papers and keynotes.

An outstanding example of the prominent participation of WP4 in top conferences in the area of network management is the last IFIP/IEEE Symposium on Integrated Network Management (IM 09) (http://www.ieee-im.org/2009/) IM is the main venue for original research in the area of network management. Its last edition was held in NY, USA in June 2009. It included technical sessions, application sessions, a mini-conference, panels, tutorials and

several co-located workshops. The contributions by 4WARD partners to IM this year include four papers in the technical sessions (out of a total of 41), a paper in the mini-conference and a poster. In addition, Dr. Marcus Brunner gave a keynote speech on In-network Management in the workshop on Management of the Future Internet (ManFI).

WP4 representatives have also met other EU-funded projects to discuss their work in network management. For instance, WP4 participated in the Joint EMANICS, AutoI, Self-Net Workshop on Autonomic Management organized by Professor George Pavlou (UCL) in April 2009. The goal of the workshop was to bring together researchers working on autonomic management in European projects. The workshop included 14 talks (including one by 4WARD) and had some 40 participants, most of them from academia.

Finally, WP4 has also disseminated some key results in the Future Internet Summer School, where Professors Danny Raz and Rolf Stadler gave a course on INM in July 2009.

4 Motivation and Scope

4.1 Technical Limitations of Existing Approaches

Traditional management solutions typically execute in management stations that are outside the network. These stations interact with the managed devices, mostly on per-device basis, in order to execute management tasks. This includes fault, configuration, accounting, performance, and security management. During network operation, for instance, a management station periodically polls individual devices in its domain for the values of local variables, such as devices counters or performance parameters. These variables are then analyzed on the management station by management applications.

This paradigm of interaction between the management system and managed system has been used by traditional management frameworks and protocols, including SNMP, TMN and OSI-SM.

This paradigm has proved fairly successful for networks of moderate size, whose states evolve slowly and thus their configuration rarely needs to be changed and no rapid interventions are required. Many of today's emerging networks depart from this model. We envision that in the future Internet, particularly with its wireless and mobile extensions, networks will include millions of network elements, whose state will be highly dynamic and whose configuration will need to adapt on a continuous basis.

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

4.2 INM Business Value Opportunities

As a precondition for their deployment, new technologies and management principles have to be associated with business opportunities by enabling new services or improved quality of service to the users and/or by reducing costs in production and operation. Most INM design principles are directly related to business value. In this section we are addressing the main impact factors of INM on capital and operational expenses (CAPEX/OPEX) in general and for some typical scenarios.

4.2.1 INM impact on CAPEX/OPEX

A shift to self-organised and self-controlled components and schemes in the management of the future Internet will have an influence on the CAPEX/OPEX distribution, where a tendency towards more automated processing results in reduced OPEX. The evaluation on the CAPEX/OPEX distribution heavily depends on the accounting scheme of each operator for large networking platforms. In addition, new networking scenarios are enabled for home or

corporate users without involving network operating staff. In particular, we identify the following impacts for the INM design:

- Variation on per-equipment CAPEX

  Self-management adds functionalities to the equipment, which are regarded as value-added functions for operators. These extra-features will reflect undoubtedly on an increment of cost in the individual equipment compared to the equipment without self-management functionalities. The bottomline here is that network equipment is nowadays becoming mature, and the basic technology is becoming cheaper. Added functionalities, like self-management, is a strategy to keep per-equipment CAPEX constant while building added-value functions to legacy equipment.

- Variation on CAPEX related to embedded network-wide INM management capabilities

  Self-management can significantly increase the exploitation of an existing infrastructure. Current approaches rely on over-provisioning policies, because existing management operations are not considered trustful for prompt resilience. INM self-management functions embedded in the network support fast recovery from failures and optimize resource efficiency like bandwidth, buffer, server capacity such that more users can be accommodated with the existing network. Offering services to the same population with less network infrastructure means CAPEX reduction in terms of investment as well as energy consumption on the OPEX side.

- Variation on OPEX

  INM can impact on the contribution to operations and maintenance (OAM) costs. Self-management automates much of the operations that today require human intervention. In this way OPEX is reduced.

- Value of the investment

  Basically this is measured through key indicators, which are defined at business level and give an estimation of the return of investment. Examples are:

  (i) service quality: out of services rate (e.g. how many sever failures occurred in a month) and other parameters defined in service level agreements (SLA); number of trouble-tickets received to operation support (the lesser, the higher the saving gained),

  (ii) resource utilization: e.g. percentage of available bandwidth that is exploited on average on peak periods,

  (iii) lifetime cycles: how long are networking components expected to be operative before they will be replaced by a new generation with improved performance and/or functions,

  (iv) time-to-market of a new service: e.g. the time needed to insert new management policies or configure new monitoring functions for it.

These indicators are very much dependent on the particular business models of each operator and differ with application scenarios. We have studied several scenarios and give a brief summary of the expected business impact of INM solutions for several cases. Self-management requires a sufficient degree of standardization, which poses the basis for open markets. 3GPP has started work items on self-organizing networks (SON) within the E-UTRAN and long term evolution (LTE) framework which also addresses self-configuring and self-optimizing network use cases and solutions [3GPP TR 36.902].

4.2.2 Case studies

INM Support for Traffic Engineering and Efficient Resource Usage

Internet traffic growth rates are estimated at 45% per year on the fixed network part and even higher on mobile broadband access platforms [MINTS]. An upgrading process of links and nodes steadily has to keep pace with the increasing traffic demands, that is embedded in a

traffic engineering concept for balancing the load in the network and achieving optimum throughput in normal operation and usually also for a set of most relevant failure scenarios. But even if optimized transport paths are (pre-)established, it is still time-consuming to detect link failures and to reconfigure the paths on detours. Therefore additional capacity is provisioned to overcome unforeseen intermediate stages after changes. INM proposes improved monitoring functions for situation awareness in short time cycles and thus reduces phases subject to unbalanced or instable load in the network.

As a consequence, over-provisioning for instable periods can be saved if they can be kept short, i.e. below the usual 50ms rerouting demand. Experience with traffic engineering in IP backbones has shown that the resource utilization can be increased by at least 20%-30% [Hass05] when fast rerouting immediately reacts to failure situations, which means a corresponding decrease in traffic driven CAPEX for bandwidth on the IP and transport layer.

The Value of Using INM within Femtocells

Femtocells are cellular access points that connect to a mobile operator's network using residential DSL or cable broadband connections. Femtocells can be used by residential users or enterprises to enhance in-door coverage and to improve QoS support. The large number of femto access points and the way of their operation, which is clearly different from that of macro access points, make it virtually impossible to use the traditional management paradigm where the management controls each access point directly and exactly. The growing field of their rollout and deployment thus presents an interesting area where INM principles and technologies can be used to address the related challenging network management issues. INM's potential to support distributed and autonomous management can be useful in the following aspects:

- Femtocell networks use a so-called Femto Access Gateway (FGW), at least for 3G HSPA, a centralized network element coordinating network access for femto installations. We assume that parts of this functionality could be executed "inside" the femto cloud such that less capacity is needed on the gateway. For 3G releases 8 and 9, i.e. the Long Term Evolution (LTE) femto radio nodes (e) HomeNodeB, the gateway network element is optional with respect to the connection to the management platform. This fact strengthens the need for self-organizing management and optimization technology [3GPP32.816, 3GPP32.821].
- Since management support for Femto Access Points (FAP) is limited, there are obvious options for improvement, for example, anti-interference management, which can be implemented according to the INM principles, i.e. mainly by letting INM self-managing entities communicate and negotiate management issues with each other. While current practices usually deploy centralized management schemes by using standards such as TR-069 [BBF69], some of the functionality can instead be distributed in an INM self-managing entity hierarchy.

Dynamic Networking Environments

The Future Internet will integrate sensor, mobile ad hoc and peer-to-peer networks as dynamic environments being subject to very frequent changes in connection topology. With increasing dynamics, network-wide routing tables and network status information in a centralized view are becoming more and more imprecise. Then distributed and self-organizing approaches remain as the only effective solutions in highly dynamic communication scenarios. INM implementations are likely to establish at first in such dynamic application fields where classical management approaches are insufficient. In a next deployment phase the integration of INM functions into network elements may be cheap and reliable enough for static large scale platforms. Systems under centralized control can also benefit from increased flexibility introduced by more intelligent network elements, e.g. for tracing of and responding to unexpected failure cases.

4.3 The INM Approach

Today's deployed management solutions have two main characteristics. First, managed devices generally present simple and low-level interfaces. Second, management stations interact directly with managed devices typically on a per-device basis. There is no interaction among managed devices for management purposes and those devices have little autonomy in making management decisions. As a consequence, managed devices are "dumb" from a management standpoint.

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

In 4WARD, we propose a clean slate solution to network management. By this, in the context of 4WARD, we mean a way of envisioning and engineering management concepts and capabilities that abandons these two characteristics.

The approach we propose in the 4WARD project is called In-Network Management (INM). Its basic enabling concepts are decentralization, self-organization, and autonomy. The idea is that management tasks are delegated from management stations outside the network to the network itself. The INM approach therefore involves embedding management intelligence in the network, or, in other words, making the network more intelligent. The managed system now executes management functions on its own. It performs, for instance, reconfiguration or self-healing in an autonomic manner. It reports results of its action to an outside management system, or it triggers exceptions if intervention from outside is needed.
In order to realize this vision, a management entity with processing and communication functions is associated with each network element or device, which, in addition to monitoring and configuring local parameters, communicates with peer entities in its proximity. The collection of these entities creates a management plane, a thin layer of management functionality inside the network that performs monitoring and control tasks.

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

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

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

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

- A high level of scalability of management systems, for instance, in terms of small execution times and low traffic overhead in large-scale systems. This will allow for effective management of large networks.

- Fast reaction in response to local and external events. This will increase the adaptability of the network to various kinds of faults, configuration changes, load changes, etc.Together with embedded functions, this will lead to with a high level of robustness of the managed system.
- A high business value for INM technologies through reducing capital and operational expenditures.

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

4.4 Scope of the Work in 4WARD

It is often useful to understand the scope of network management by using the functional model defined by ITU-T and ISO, which partitions network management into five functional areas (referred to by the acronym FCAPS): Fault and Configuration Management, Accounting Management & User Administration, Performance and Security Management. A somewhat broader definition of network management can be found in [Clem06], which says that "Network Management refers to the activities, methods, procedures, and tools that pertain to the operation, administration, maintenance and provisioning of networked systems." It is clear that 4WARD cannot provide a comprehensive treatment of all these aspects, but must focus on key components that are relevant to developing and evaluating the INM paradigm.

From the five functional areas, we have concentrated on aspects of performance and fault management (Chapters 6 and 7). Regarding the definition in [Clem06], we have given weight to technical contributions in support of operation (Chapters 6 and 7), while, at the same time, developing the INM framework (Chapter 5) with the vision that it can ultimately support all aspects mentioned in [Clem06].

5 INM Framework

The framework for in-network management (INM framework) is the enabler of INM (see Section 4.3) in general and management functions (see Chapter 6 and 7) in particular. Making use of its features will allow to overcome the technical limitations of today's prevalent management systems (see Section 4.1) and lead to novel business values as described in Section 4.2. Based on the instruments for a clean slate design of INM defined in [D4.2], the INM framework follows four principal guidelines, which together capture the structural and functional nature of INM on one side, and a distinction into organizational and collaborative management tasks on the other side:

- *Co-location* (structural): The INM framework enables the realization of management functions that are *co-located* with the services subject to management. This objective emphasizes INM's tight integration from a *structural* perspective.
- *Co-design* (functional): Complementary to co-location, the INM framework adopts a style of designing management functions in conjunction with service functions, termed *co-design*, which emphasizes INM's tight integration from a *functional* perspective.
- *Collaboration* (functional, collaboration): In order to achieve highly distributed and self-adaptive operation of management functions, the INM framework defines how *collaboration* between basic building blocks of management functions, called management capabilities, is to be performed.
- *Management by objective* (functional, organization): The INM framework follows the paradigm of *management by objective* rather than that of *managed objects*, defining how to enforce and report high-level objectives in a hierarchical manner by means of management domains, self-managing entities, and management capabilities.

The INM framework does not define the details of how the functional aspects are achieved, such as the enforcement of high-level objectives or the particular way different management algorithms will interact. Rather, the framework defines architectural elements, their properties, and interaction, according to which management functions shall be integrated.

Section 5.1 provides an overview of the INM framework's main concepts. In Section 5.2, the framework's architectural elements and their relations with one another are described in more detail. Section 5.3 describes how algorithms are integrated with the INM framework. Section 5.4 provides a more detailed account on how information shall be stored and accessed under the INM paradigm. Section 5.5 describes the dynamics of INM around the notion of knowledge-supported workflows. Section 5.6 discusses two concrete alternatives of how to implement INM in real systems. Finally, the conclusion in Section 5.7 closes this chapter.

5.1 Framework Overview

Figure 5-1 illustrates the INM framework's main concepts. On the operator side, the *global management point* (GMP) is the high-level entry point into the management of a network according to 4WARD's architectural framework [D2.3]. The GMP is the management interface to the operator, providing the highest level of abstraction by means of objectives.

In the first hierarchical refinement, the global management point provides access to one or more *management domains*, each allowing access to a well-defined subset of the embedded management functions. A management domain is also the main abstraction that enforces security by restricting access to only those management functions that an operator is allowed to handle. Multiple domains may exist at any time during network operation, their configuration may change and they may be set up and torn down dynamically over time. Subdividing the embedded management plane of a network may occur in both structural and

functional terms, such as a subdivision by self-managing entities and the isolation of only performance-related management functions, respectively.

While the purpose of management domains is to identify a well-defined management subject in terms of structural and functional characteristics, *self-managing entities* (SEs) are service-oriented and encapsulate self-management functions of individual services, such as NetInf [D6.2]. SEs collaborate with one another and enforce objectives on the service level in order to meet service-specific objectives dictated by the operator's high-level objectives.

Each self-managing entity contains multiple management capabilities (MCs), which implement the actual self-* (e.g. self-adaptive) management algorithms on a fine granularity. All algorithms described in Chapter 6 and 7 are implemented at this level. While containment of MCs within SEs is one possible realization, MCs may also exist stand-alone. This configuration is useful for MCs that encapsulate more generic management algorithms, for example, aggregation algorithms for monitoring according to Chapter 6.

Each management domain combines a number of self-managing entities, more precisely, a structural or functional subset of a set of SEs. In other words, a management domain *slices* a set of SEs in such a way that a limited management functionality is isolated. In particular, slices may correspond to any of the self-* properties defined by an SE as described in Section 5.2.2. Slices are indicated in Figure 5-1 by multiple overlay planes intersecting the same set of self-managing entities.

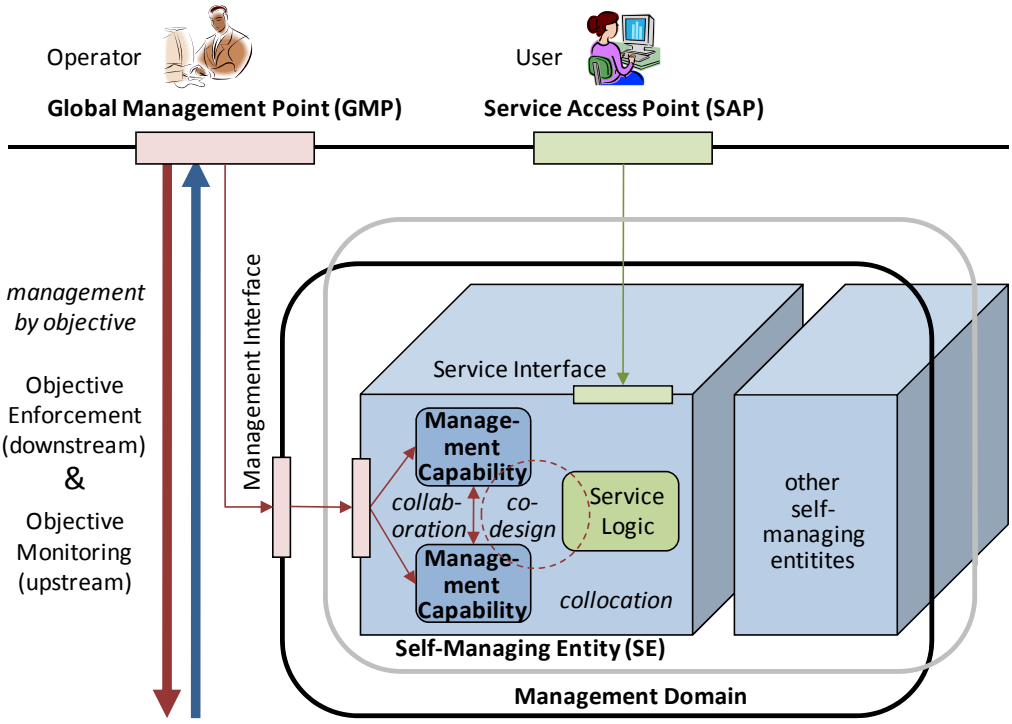


Figure 5-1: Overview of the INM Framework

The enforcement and monitoring of objectives occurs along the hierarchy of the previously described elements. Regarding the enforcement (downstream), each objective is specified on an abstract level by the operator at the level of the global management point and split into sub-objectives via management domains and self-managing entities until reaching individual management capabilities. On the other hand, objectives are monitored and aggregated towards the global management point (upstream) in the opposite sequence of

elements. We note that the handling of objectives in either downstream and upstream direction is nontrivial, and we provide additional notes on this problem in Section 5.7.

Figure 5-1 also shows how each of the four principal guidelines is considered in the INM framework. In particular, co-location and co-design operate between management and service functions, the latter being accessible via service access points (SAPs). More details on co-design techniques are discussed in Annex B and [Dud09].

5.2 Architectural Elements

The INM framework defines a concise set of architectural elements from which any distributed and embedded management structure can be created. This section describes these elements and their relation with one another in terms of interfaces in more detail.

In order to describe and instantiate frameworks that are specific to the network management domain, we make use of a meta-framework, which is based on the concept of meta-models that are widely applied when designing IT systems [Mof02]. Figure 5-2 provides a view of the components and their interrelations of the meta-framework of INM, using the Unified Modelling Language (UML) notation.

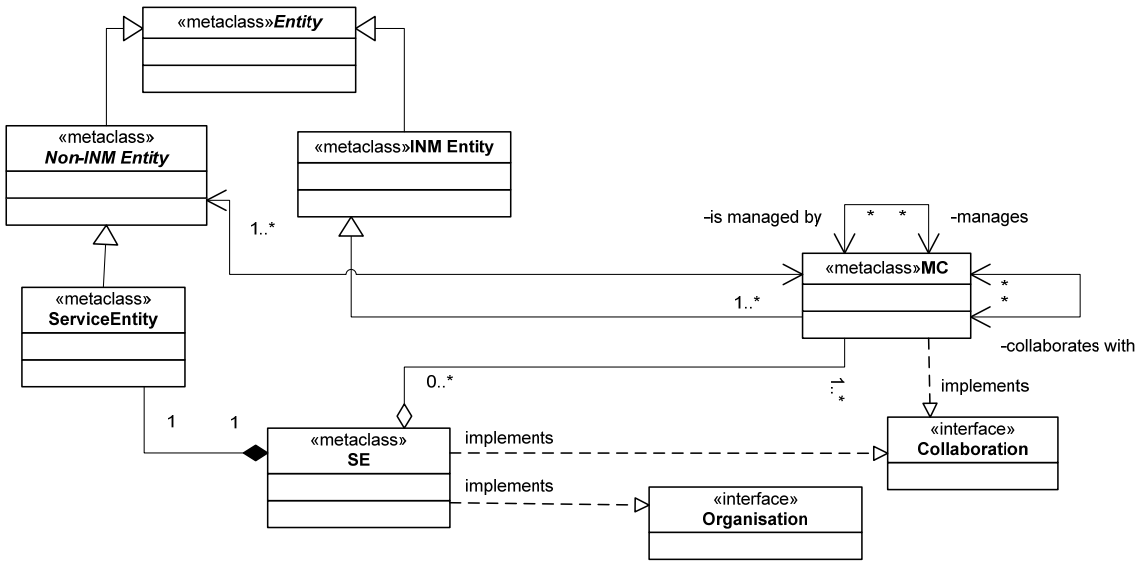


Figure 5-2: Model of the INM Meta-Framework

This is a system-wide view which identifies the categories of entities to represent a specific part or component of the system. The entities are split into two types, INM entities and non-INM entities. This separation was made in order to support migration from current Internet paradigms (where non-INM entities are simply to be managed by management capabilities) to a future Internet where self-managing entities (SE) embed management capabilities according to the degrees of embedding defined in [D4.2]. Service entities are non-INM entities that encapsulate system services. Our definition of a system service comprises services related to functionality traditionally allocated to all the layers of the TMN model [ITU00]. Management capabilities (MCs) and self-managing entities (SEs) and their relation with each other is described in more detail in Section 5.2.1 and 5.2.2.

The organization and collaboration interfaces are also key to the interaction of the different kinds of architectural elements. Figure 5-3 provides a perspective on the INM framework that illustrates information flow along these interfaces. By convention, management functions pertaining to objective enforcement and objective monitoring on one side and to the collaboration between management functions on the other side are mapped to organization

and collaboration interfaces, respectively. Figure 5-3 shows how collaboration is contained within a management domain and only between MCs and SEs of the domain.

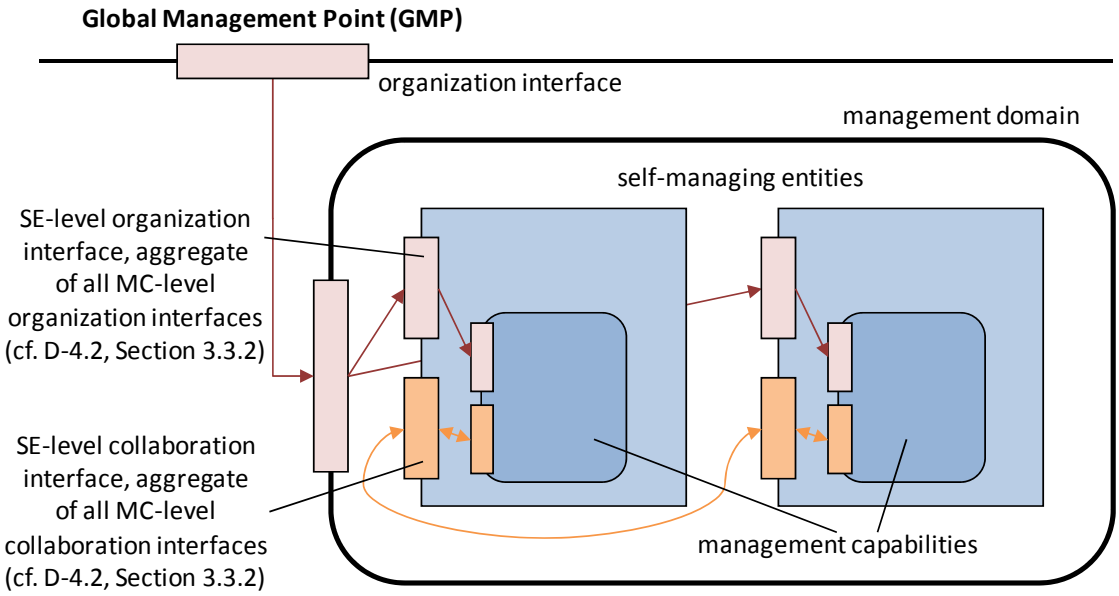


Figure 5-3: Collaboration and Organization Interfaces

5.2.1 Management Capabilities

Embedded management capabilities (MCs) encapsulate all management algorithms that are implemented in line with the INM approach. They implement the collaboration interface (see [D4.2] and Figure 5-2) in order to communicate with each other for the purpose of fulfilling management tasks, and the organization interface to handle objective enforcement and reporting. Furthermore, MCs typically follow an internal structure that encapsulates the mapping from organization to collaboration aspects (However, the internal structure is up to the implementation of the MC and the INM framework only provides a rough guideline in this respect). Figure 5-4 depicts the typical structure of and interaction between management capabilities.

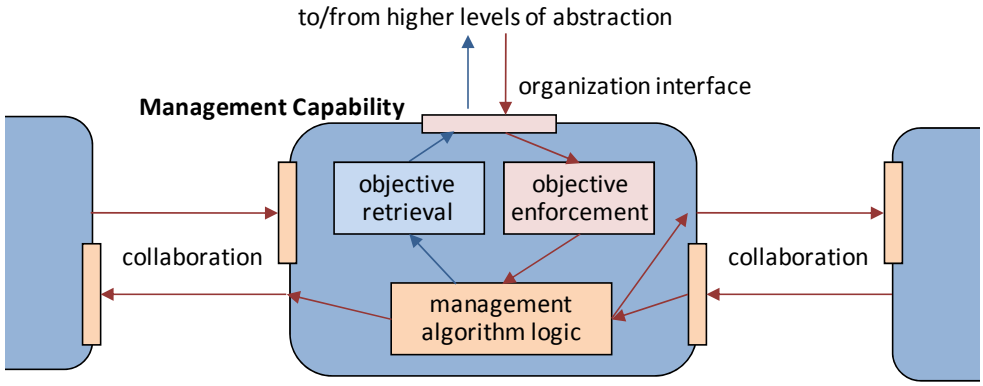


Figure 5-4: Management capabilities: structure and interaction

Management algorithms are modelled within a management algorithm block of an MC. The management algorithm interacts with other MCs via the collaboration interface. The particular implementation of this interface may follow different styles, for instance, an RPC or

REST style, discussed in more detail in Section 5.6). A typical interaction of an aggregation algorithm, for instance, may be the exchanging of partly aggregated values.

Objectives are handled by management capabilities in the following way. For objective retrieval and composition in upstream direction towards the high-level objective visible at the global management point (see Figure 5-1), the objective retrieval block (Figure 5-4) is responsible for accessing a low-level objective's value. Upon retrieval, it is handed via the organization interface upstream to other management capabilities. For objective enforcement, the self-adaptation block retrieves objectives via the MC's organization interface and refines them until the management algorithm is appropriately parameterized via a low-level objective. Not shown in the figure is interactions between MCs that occur via the organization interface, such as the refinement of objectives between two self-adaptation blocks. Self-adaptation is described in more detail in Chapter 7

A typical interaction is the case in which a low-level objective is violated in a management capability and resolved by the next upstream capability using a specific self-adaptation mechanism without the need to report to the GMP. This benefit is key to INM, where objective violations are managed as far as possible in an embedded way, local to the occurrence of the violation. Explicit interaction by an operator is then only required when the violation cannot be handled appropriately by available self-adaptation logic in an autonomous way.

Management capabilities serve the specific purpose to *terminate* any upstream/downstream embedded management hierarchy at the bottom. For instance, in any upstream configuration of management capabilities exist some capabilities that connect to the lowest level of managed entity. The most important types of management capabilities in this view are the ones that interface with the hardware. For that purpose, the INM framework specifically considers management capabilities that interact at a very basic level with protocol layers that are close to the communication drivers. This is illustrated in the following by the cross-layer QoS (CLQ) management capability designed in detail in [D4.2].

The CLQ has a particular feature compared to other MCs, i.e. it needs two types of protocol stacks (called Netlets in [D2.2]) as shown in Figure 5-5. One type of stack has actually no protocols included for communicating directly with the hardware. This is requested by three major beneficiaries: a) the bottom-up approach while performing periodical measurements on top of the MAC sublayer; b) the top-down approach, in case of emergencies, performs global resource allocation by replacing the Generic Path management that failed; c) a hop-by-hop transport between nodes. The Netlet type 2 may include legacy/future protocols and it is needed for exchange of information with other MCs located in different nodes. Examples of algorithms employed by Cross-Layer QoS for self-adaptation are anomaly detection (AD) and "Not All aT Once!" (NATO!) presented in Chapter 6.

As additional service, the measurements could be done by request. Thus, the AD module sends a request to obtain current link latency/round-trip-time via one end-to-end transaction. The MC receives the request and immediately sends the probe. If it obtains a probe reply, the measured latency is sent back to the AD-module. The AD-model autonomously adapts to the new observation. In case a probe reply is not received, the CLQ does nothing, but the failed probe response will be accounted for in the AD-model.

An example of getting benefits from INM algorithm is NATO!. A strategic node needs to get information (i.e. available transfer rate, one-way delay, BER etc.) from all nodes within its management domain. This could be provided by the CLQ being present in each node. However, this approach is not always realistic, because some of the nodes are too simple to integrate sophisticated mechanisms. It may lead to either too much traffic for information exchanges between nodes, or to limitations of scalability. With the NATO! algorithm, a strategic node could interrogate all the devices within its domain with a specific request (i.e. what nodes are able to accept flows with a given transfer rate and/or one-way delay). If

statistics show that the number of nodes fulfilling the requirements is large enough, CLQ could involve QoS-aware routing only. If the statistics are bad, then network coding is an alternative mechanism. To conclude, NATO! could help INM to decide on whether to make use of either QoS-aware routing or network coding [Pol09].

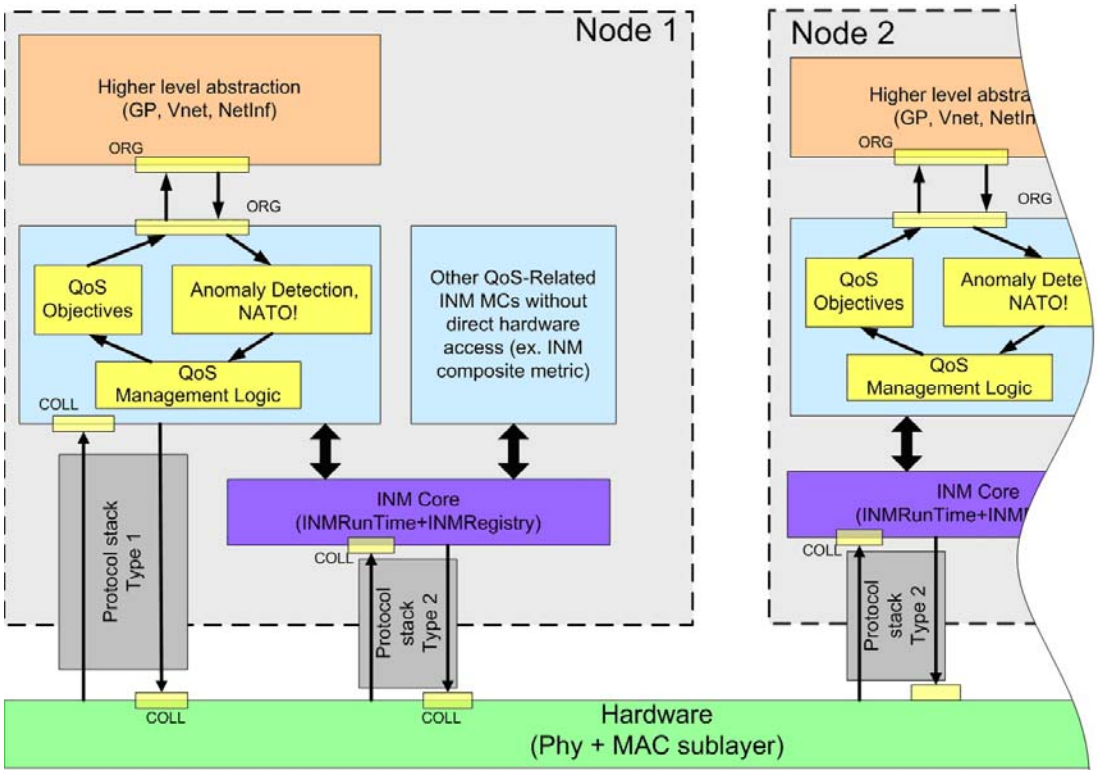


Figure 5-5: Examples of interactions between a specific INM MC (i.e. Cross-Layer QoS) and other management capabilities and algorithms

5.2.2 Self-Managing Entities

Self-managing entities (SEs) are containers for the functionality required to build self-managing services and allow to create a complete service hierarchy spanning all TMN layers. They expose to the outside world the *organisation* and *collaboration* interfaces, which are realised by sets of MCs (see Section 3.3.2 in [D4.2]) and encompass the properties necessary to achieve autonomous operation of a future Internet service-centric network infrastructure with only high-level intervention from the operator, as previously illustrated in Figure 5-2. The organisation interface contains all the exposed capabilities that allow for the transmission and negotiation of service goals, assembling entities as result of composition and reporting on the conditions of service delivery.

Figure 5-6 provides a more detailed account of the SE structure, expressed in an UML description. The Service Level Agreement (SLA) allows the SE to be the enabler of the management-by-objective paradigm (see also to the beginning of Chapter 5). Introduced in 4WARD with [D4.2], this concept calls for using objectives, expressed as high-level descriptions, as the means to steer the behaviour of a system. Objectives are hierarchical between SEs over the organisation interface. In terms of management functionality, objectives could be related to how efficient an algorithm should run (in terms of bandwidth consumption for a monitoring algorithm, for example), or availability in case of a connectivity service. An SE must have at least one SLA associated to the objectives directly related to the service performed. The options specified in the SLA are to be decomposed into management tasks and requirements which will in turn be assigned to MCs along the downstream hierarchy.

The Capability Level Statement (CLS) is a declaration by an SE on what it is able to provide in terms of services, along with a quantitative reference to the available abilities. All requests for services from SEs to other SEs are associated with the CLS and an SLA. Once the SLA is negotiated and agreed between the two SEs, the serving SE changes its CLS statement to reflect the currently available capacity to serve other SEs. The CLS thus changes dynamically when an SE initiates or terminates a service offering to another SE. The Service Access Point (SAP) interface presents the service to the outside world along well-established SOA (Service-Oriented Architecture) principles (also see Figure 5-2).

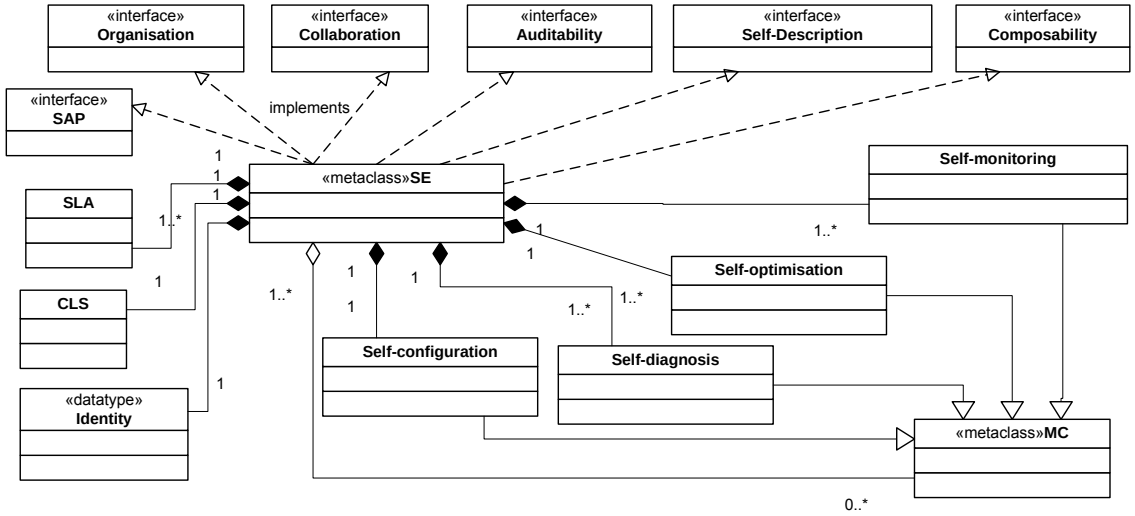


Figure 5-6: Structure of a self-managing entity.

D-4.2 of 4WARD [D4.2] refers to the capabilities associated with the self-* properties. These properties enable migration to a more agile, widely distributed paradigm for network management. Kephard and Chess [Kep03] formulate four aspects to be considered for achieving self-management in an autonomic computing scenario: self-configuration, self-optimization, self-healing, and self-protection. The SE properties take this work into account and add other aspects of the meta-framework (such as composability and auditability) to enable wide-scale decentralisation of network management functionality.

While the SE is part of a meta-model (an abstract logical construct), one SE can be distributed among multiple physical locations at runtime. The instances part of the same SE would use a collaboration interface to realise the self-monitoring functionality at an SE level, for example. They would be able to find each other and attach to one another through the organisation interface. An SE contains management capabilities to refine the goals received over the organisation interface. Policies may be used to guide this process towards solutions that are considered optimal for particular systems.

5.2.3 Creation and Usage of Management Domains

The basic architectural elements discussed in the previous sections can in principle build up a highly distributed flat architecture, where MCs can be discovered and executed on a peer-to-peer basis. However, network management functions can be more easily and better coped with if some organizational or logical structures are applied. The traditional way of network management is to manage groups of devices using an element manager, while several of these are controlled by a domain manager, and on top of everything, the actual network management. The state of the art is represented by larger, umbrella-like network management systems that cover several smaller systems. Network management is used to create or modify

network structures, and mostly this involves human operators to perform the separation of node sets. Such structures are usually of static and hierarchical nature.

In the context of INM, the notion of management domain is flexibly used to structure INM functions. In the simplest case, a domain is a group of self-managing entities. It implies the physical boundary or the logical scope within which the SEs execute their functions or assert their control. Domains in INM can be static or dynamic during their lifetime, and SEs can belong to several domains at the same time, each domain having a different type.

The use of management domains serves to meet many requirements, including:

- Administrative issues, e.g. physical possession (ownership) of subsets of nodes by different shareholders.
- Location areas, e.g. limiting some INM algorithms to a certain number of nodes, or hops or geographical distance.
- Limited scopes of management operations which are valid only for e.g. certain regions, access technologies, handsets or home networks.
- Virtual network operation, e.g. limit the scope for certain regions, access technologies or network resources in general.
- Service specific, performance or security related requirements.

Creating a domain can be performed by either outside-in or inside-out methods (Figure 5-7). The outside-in method is usually adopted for domains derived from an operator's objective, and can therefore be configured either statically beforehand or dynamically through the global management point (GMP, see Figure 5-1). The inside-out method is usually adopted for management operations coupled with the network functions and it creates domains between cooperating neighbouring nodes which group themselves subsequently into larger conglomerates. Such cooperative groups can be formed, for instance, based on neighbourhood abstractions, which are in turn based on network topology. The co-existence of the two methods enables scalability and guarantees a high level of flexibility in the aggregation of management operations. Especially the inside-out method enables the separation of potentially extremely large number of nodes into well organized and easily manageable sets, adhering to the INM paradigm with self-organizing behaviour.

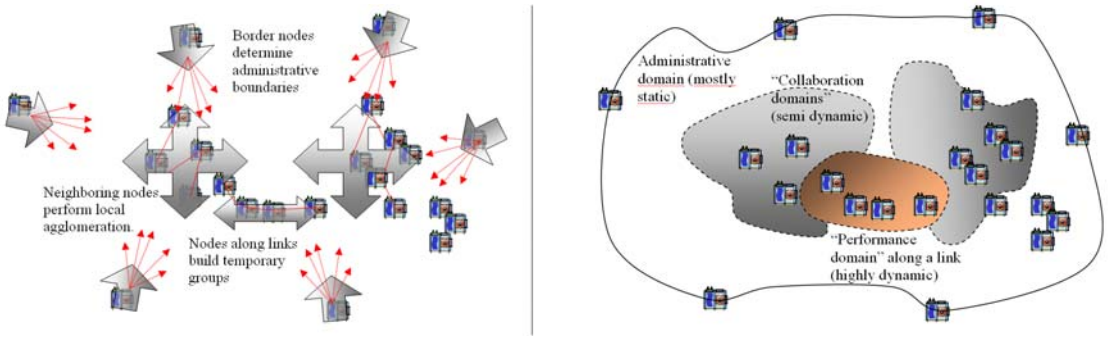


Figure 5-7: Outside-in and inside-out methods of initiating and forming domains (left: preparation signalling; right: established domains)

Creating domains can be achieved by several technical means, including:

- Gossip-based algorithms can be adopted in networks with no configuration retrieved from the operator or services;
- Algorithmic creation of domains without or with limited central control can be performed using self-organizing algorithms;
- Graph-colouring and/or cellular automata taken from computer science fields;

- Algorithms for finding dynamically a master node within a set of nodes;
- Mechanisms in line with virtual networks (Vnet) [D3.1.1] and compartments [D5.2].

A domain, initiated and formed by either the outside-in or inside-out method can be enforced into the management capabilities at different levels, either as information elements encoded in the network nodes or soft-state, like a “scope” for specific management functions.

5.3 Integration of Algorithms into the INM Framework

In this section we demonstrate how the INM algorithms developed in Chapter 6 and 7 can be integrated into the INM framework and its components. In this process, the algorithm developer must consider three main aspects:

1. How to model their algorithm using MCs and how many MCs to use;

2. What functionality to publish via the MC organization and collaboration interfaces. As part of this, the objectives which the MC will be able to enforce must be specified.

3. How the algorithm will interact with the INM environment (Chapter 8). The developer must be aware how to deploy, register, discover and use publish/subscribe functionality provided by the INM environment.

With respect to modelling an algorithm the key architectural component to implementing a management algorithm is the management capability. The algorithm designer may use one or more MCs to realise their algorithm. The decision on how many MCs to use is dependent on the type of algorithm being developed, its functional parts and complexity.

We use the Generic Aggregation Protocol (GAP) as an example here (Figure 5-8). GAP is a protocol for continuous monitoring of network state variables (also called aggregates). Aggregates are computed from device counters using aggregation functions, such as SUM, AVERAGE and MAX. Sample aggregates 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.

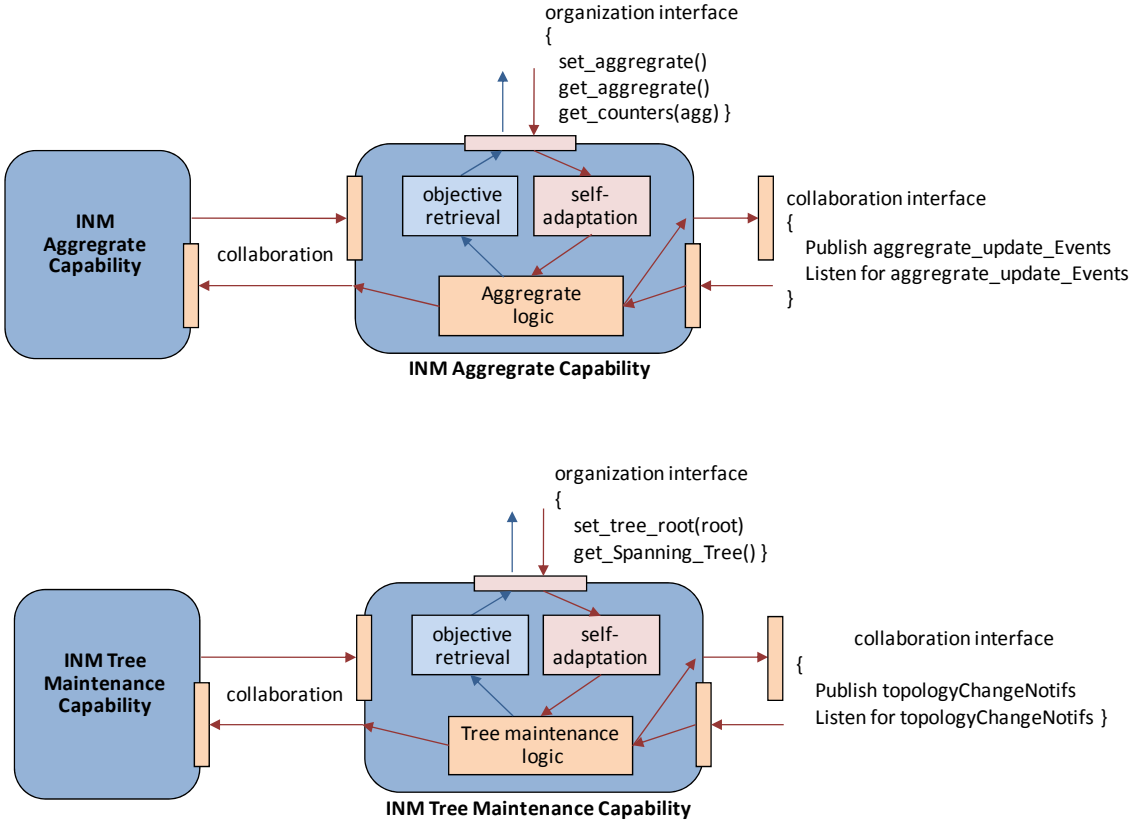


Figure 5-8: Sample GAP MC Collaboration and Organization Interfaces

The first step is to decide how many MCs will be used to model the algorithm. This is a design choice and therefore the designer has some flexibility in this step. That said, the choice should be made taking into consideration aspects such as software modularity and what interfaces the designers want to make available to third parties. For the case of GAP, we have decided to use two management capabilities. The first one, “Tree Maintenance”, is responsible for creating and maintaining an aggregation tree, which is instrumental in the distributed computation of the aggregate. The second MC is called “Aggregator” and it is responsible for computing the aggregate, taking as input local counters.

The second step is for the designer to specify the interfaces for each MC, from our GAP example, the ‘Tree Maintenance’ and ‘Aggregator’. Sample interfaces for these MCs are shown in Figure 5-8) When GAP runs, the first message exchanges are between instances of the “Tree Maintenance” 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 are among “Aggregator” MCs on different nodes. They also use the collaboration interface to compute network state variables through incremental aggregation. The consumer of the monitoring data may specify the aggregate he/she is interested in through the organisation interface of the 'Aggregator' MC.

The third main point of integration for the algorithm developer is how their MC or set of MCs interact with the INM environment. The INM environment provides a runtime which abstracts itself from the underlying technology choice and provides functionality for deployment, registration, discovery of MCs and also publish / subscribe functionality.

5.4 Management Information Storage and Access

Traditionally, network management is logically and also location-wise fairly centralized, retrieved from the network through some management protocols and typically stored in a

central database. Also the NM system is in charge of retrieving the information from the network, processing it and storing it in the database. Since the management information source can be any network element, a decentralized storage may adapt better to the decentralized nature of the information, especially if the information is mainly needed and processed locally. The expected improvement is the storage of INM data, information and potentially state in a way that has benefits in reliability, access speed, and query-like search options. The decentralization of management information storage and access will also work in line with the decentralization of the management functionality itself.

We consider two different approaches for storing and accessing INM management information in a decentralized way. The first one is WP6's Network of Information (NetInf) approach to information storage and handling [B08], which is based on separating content and locator information and applying one or multiple indirection steps, resulting in the ability to store and retrieve information objects in a more generic way. NetInf is foreseen to provide a storage middleware allowing to store information elements in a decentralized way and provide search and retrieval capabilities.

The second approach is the use of the situation awareness framework [D4.2] which covers the management of context and of pre-processing of this information into an assumption about the situation of an entity. The aggregation of context into situation is already part of the decision process, by narrowing monitored information to a more abstract level for the decision. Each entity, INM application or any component that offers INM information registers itself to the directory indicating the sort and location of the information offered. Vice-versa, each entity, application or component is able to retrieve needed INM data by sending a resolve to the directory which replies with the correspondent location of the requested information. Using the location information, INM data can be requested directly P2P afterwards, which can be requested either by a get or subscribe. In the former case the INM data will be sent just once, in the latter one it will be sent continuously every time new data is available.

The approaches reside on different levels: NetInf is a middleware, while the situation awareness framework is placed below on the level of network functions. The approaches can be used in highly dynamic environments to cope with different time constraints. Short-term information can be exchanged directly between involved components via the situation awareness framework. Long-term information can be stored in NetInf for wider distribution or later usage. Short-term information will not be stored in a platform with its own mechanisms for distribution and management. A key concept for interworking is the aggregation towards more abstract and long-lived information.

Both information management approaches are related to the usage and creation of domains (cf Section 5.2.3), which help to structure the INM regarding functional or spatial issues. To place or access a directory component of the situation awareness framework, an existing domain with its name resolution can be used. The same holds true for information access and distribution via NetInf for INM. Further considerations on the storage requirements for management information are made in Chapter 9.

5.5 Knowledge-Supported Workflows for Self-Organisation INM Processes

The self-organisation of INM processes is essential to the INM paradigm in that it makes INM operate in a semi-automated distributed way. In Section 3.5 in [D4.2] we have defined the dynamics of INM processes around the notion of lifecycle of self-managing entities. We extend and consolidate this approach in the following using a workflow-based approach that illustrates how complex INM processes are executed from design to operation time.

It is essential for the human operator to be able to design and deploy the self-managed system based on business-level goals (that is, high-level objectives via the GMP, see Figure 5-1 and to assert the control over the automated system in case of need. An enabling

approach is the deployment of *knowledge-supported workflows*, which dynamically glue and control the INM processes as exemplified in Figure 5-9.

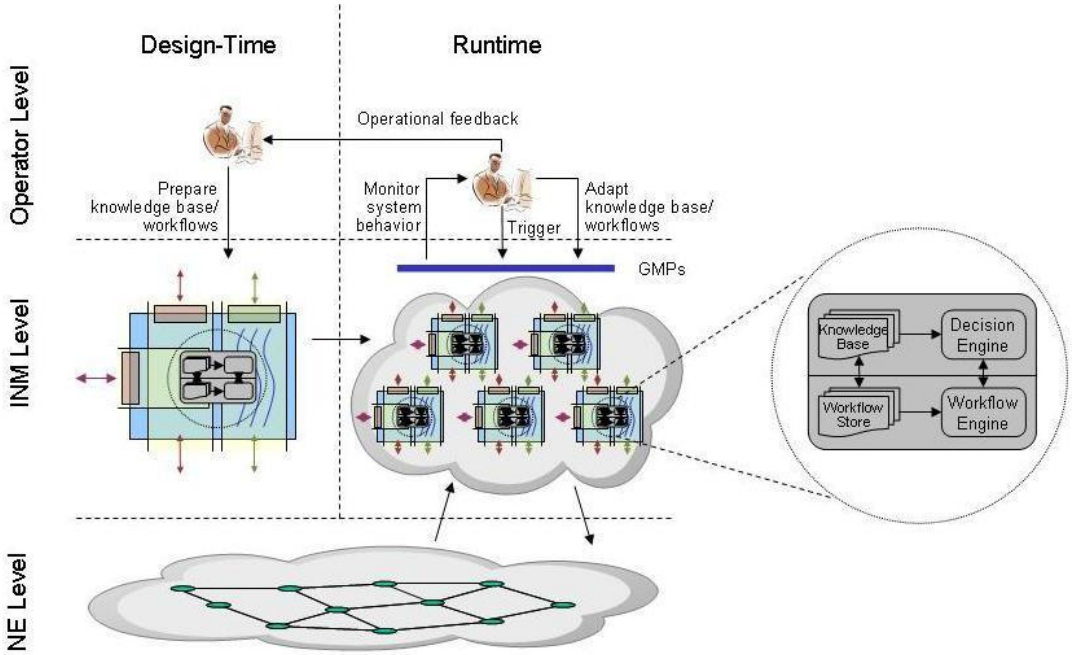


Figure 5-9: Control of INM processes/ flows by knowledge-supported workflows

Basically, the operator analyses the relevant use cases and defines tasks to be executed to reach the intended business goals at design time. The variations in the task sequences or in the parameter settings are identified as well in order to cope with changing context and situation. Together with error handling procedures, these "goals" and "intelligence" are formulated in the form of rules and workflows, which are applied by the system at run time.

During run time, the decision engine reacts on incoming events and triggers the execution of appropriate workflows. Since the decisions and the branching of the workflows are dependent on dynamic context and situation information, some forms of self-adaptation are achieved by the automated management system. Normally the operator just needs to monitor the functioning of the system and to identify possible problems or improvement potentials, instead of "operating" the network step by step. The operator can introduce necessary changes in the system behaviour by changing and updating related knowledge bases and workflow stores, without the need to change other components or implementations.

In the INM framework, knowledge-supported workflows can be used for realizing management processes and service processes. A group of self-managing entities can cooperate loosely with each other by deploying identical or similar policies/rules and workflows for common purposes. Also, a lead SE (a pre-configured or dynamically elected "master") can coordinate the procedures and flows among the INM entities in a tightly cooperating mode.

As to the realization of the above concepts, it is expected that different forms of knowledge support and workflows are feasible for different cases. An experimental system for self-organizing mobile networks is currently being built with policy-oriented control [Rom07], [3GPP 23.203] and Java-based workflow engine [JSN09], [OMG09]. A management use case of mobile networks called *hardware to site mapping* is used to show the feasibility of the related concepts.

5.6 Realization Options for INM

INM being distributed pieces of embedded management functionality throughout the network leads to a number of options with respect to technologies or styles of implementation. Two competing styles were investigated during the prototyping phase; a REST based approach and an RPC based approach. Both approaches are discussed below and recommendations are provided based on the prototyping experiences (see Chapter 8).

5.6.1 Approach: Remote Procedure Call (RPC)

Remote Procedure Calls (RPC) is a technique for constructing distributed, client-server based applications. It is based on extending the notion of conventional, or local procedure calling, so that the called procedure need not exist in the same address space as the calling procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them. By using RPC, programmers of distributed applications avoid the details of the interface with the network. The transport independence of RPC isolates the application from the physical and logical elements of the data communications mechanism and allows the application to use a variety of transports.

The RPC style also has benefits when used to realise INM:

- The separation of the interface types collaboration and organization is clearly distinguished by the RPC style;
- A large number of enterprise developers are very familiar with this style.

In particular, the RPC style has shown to be suitable for management capabilities, which encapsulate the management algorithms of Chapter 6 and 7. These algorithms can be classified in a way that very much follows typical object hierarchies. For instance, aggregation can be seen as a high-level type of management algorithm, while real-time aggregation, possessing a number of additional, real-time specific features, can be naturally seen as a subclass of the more generic type.

5.6.2 Approach: Representational State Transfer (REST)

Representational State Transfer or REST [Fie00] is a set of design criteria for distributed systems that stress component interaction and scalability. It is not a specific architecture but more a set of principles which one should adhere to make their architecture RESTful. REST's key principle is the role played by resources (in our scope, network management resources), which can be any entity considered significant and uniquely identifiable. Communication is stateless, i.e. there is no opening a session and keeping state between resources. Instead each request contains enough information for the receiving resource to process it.

When applying REST to INM, all management capabilities and self-managing entities become RESTful. This implies that these entities have their own set of resources, all of which can be accessed through a common interface. Systems that conform to the uniform interface constraint gain several advantages:

- System resources adhere to the same semantics for each operation in the uniform interface, thus simplifying client or peer applications by eliminating the need for custom code to support specialized interface semantics. An INM entity interacts in a similar fashion with any other INM entity irrespective of who developed them.
- Developing resources means designing an implementation to fulfil the uniform interface and its expected semantics, essentially eliminating the development phase required for designing separate interfaces for each resource, with their specialized semantics and implied workflow. INM Entity design is simplified.
- Error handling is typically a source of significant variance between interfaces as interface designers individually cook up their own data structures and exceptions for

reporting problems. Under the uniform interface constraint, however, error handling also gains uniformity.

- Without the presence of numerous specialized interfaces, overall system simplicity increases, which typically decreases the number of defects.
- INM Entities are 'engineered for serendipity' [Vin08a], i.e. INM Entities by adhering to a uniform interface increase the chances of some application in the future making use of them in such a way that was never initially envisaged, e.g. Web mashups, in which the capabilities of unrelated Web sites are combined to create new sites that provide benefits beyond those that the original developers had intended or even considered.

Some disadvantages from the using the REST style to realise INM:

- Client-Server pull approach, clients pull resource representations from servers. The ability of INM Entities to subscribe to events from other INM Entities does not exist. Support for this publish/subscribe functionality would have to be built in.
- Sending resource representations typically means sending more data with each call than in RPC-oriented systems. Even though RESTful systems are often simpler and more efficient than their non-REST counterparts, this extra data overhead can sometimes cause efficiency problems [Vin08b].

In a REST style architecture, Functionality of e.g. self-managing entities is implemented as URI addressable resources, using GET, POST, PUT and DELETE methods of HTTP to interact with them and manipulate their properties. SEs provide a set of GET, POST, PUT and DELETE methods on their organization and collaboration interfaces to allow access to their properties like self-knowledge, self-management, and auditability. Each of the properties shall be considered as a resource and will be addressable. For example, the self-knowledge property can be made available at http://se1/selfknowledge or a similar address.

5.6.3 Comparison RPC/REST Based on Value towards INM

The value which INM gains from using both technologies is summarised in the table below:

INM Considerations	RPC	REST
Ease of use from a developer perspective	RPC is widely used and majority of developers have competence in the area. Speed of prototyping is increased as learning curve is not steep.	Not as widely used as RPC but the uniform interface does ease integration for developers. Speed of prototyping is decreased as there is a learning curve.
Potential reuse of developed INM Entities	Reuse is possible but developer will need to know interfaces	REST promotes reuse as uniform interface facilitates this
Publish/Subscribe functionality	Easily provided through RPC	REST is based on polling and functionality would need to be added to support publish/subscribe.
INM Entity Interaction	Through specified interfaces. Supports multiple interfaces, e.g. organisation, collaboration	Through a uniform interface.
Scalability of INM Entities in a distributed environment	Supported	Supported and proven as the current WWW is REST based.

Amount of data transferred in each call	RPC allows the minimum in that one specific parameter from an object can be specified	With REST sending resource representations typically means sending more data in each call than RPC

Based on the different INM considerations listed on the above table, WP4 decided to follow the RPC approach for the INM prototype. There is no one correct technology to use but one must consider the distributed nature of the environment where INM will be deployed.

5.7 Conclusion

In this chapter we have described the design of the INM framework that defines how management functions are to be embedded in the future Internet based on the INM approach. Three main elements, namely, management capabilities, self-managing entities, and management domains were defined, from which complex management functions can be constructed according to co-location and co-design guidelines. In conjunction with the results in Chapter 3 in [D4.2], this work concludes the INM framework design.

Defining a management framework is an ambitious goal and not all aspects could be addressed exhaustively. In the present work, this limitation concerns mostly the handling of objectives, which is a topic of significant complexity that will require further investigation. At this point, the prototype implementation (see Chapter 8) already contains key features that allow to enforce and monitor objectives (see Figure 5-1) and to induce self-adaptive behaviour in collaborating management capabilities. These features capture the complete flow of objectives between the global management point and low-level management capabilities in a basic way, but leave space for more elaborate to-be-investigated forms of interaction to achieve more complex self-* management functions.

6 INM Situation Awareness

6.1 Introduction

One can consider a network management system as executing a closed-loop control cycle, whereby the network state is estimated on a continuous basis, and, based on this estimation, a process dynamically determines a set of actions that are executed on the network in order to achieve operational objectives. In this chapter, we consider the state estimation part of the control cycle (i.e., situation awareness). Chapter 7 focuses on the action (i.e., adaptation) part of the cycle.

Situation awareness is a very wide field. Clearly, we can not cover it completely. We have selected functionality that we regard important and challenging for the management of the future Internet. Specifically, Chapter 6 discusses our work on real-time monitoring of network-wide metrics, group size estimation, topology discovery, data search and anomaly detection.

We have engineered and evaluated a number of algorithms that provide the network state in real-time in a distributed fashion. These algorithms provide the necessary input to the self-adaptation mechanisms presented in Chapter 7.

Section 6.2 discusses algorithms aspects we have investigated in the context of distributed network monitoring. We present our conclusions regarding the performance of different types of algorithms for this management task. In addition, we outline our work on modelling the monitoring of network-wide metrics under churn.

Section 6.3 presents the different algorithms we have engineered according to their functionality. This includes detecting threshold crossings of network-wide metrics (Section 6.3.1), group size estimation (Section 6.3.2), topology discovery (Section 6.3.3) data search

(Section 6.3.4), and anomaly detection (Section 6.3.5). Section 6.3.6 outlines our work in secure computation of network-wide metrics. We briefly comment on reputation systems in Section 6.3.7.

To the functionality list above, we add that offered by the algorithms developed during the first year of the project, presented already in deliverable [D4.2]. Those algorithms provide:

- Tree-based and gossip-based aggregation in the context of MANET's
- Tree-based aggregation of histograms
- Tree-based aggregation with controllable accuracy

Finally, Section 6.4 concludes the chapter.

6.2 Algorithmic Aspects

An important aspect in the context of INM is the evaluation of different algorithmic approaches to distributed aggregation and their comparison, according to a number of key parameters. These parameters include:

- Performance under realistic, "normal" operating conditions, in the sense of accuracy and response time
- Controllability, in the sense of the ability to offset performance against overhead in a predictable manner
- Scalability. The parameter concerns the ability to the algorithm to network configurations of increasing size, without introducing unmanageable bottlenecks. Normally, this is interpreted as a requirement for sublinear growth in overhead as a function of network size.
- Robustness. The algorithm should be able to maintain functionality even under adverse operating conditions, including random faults (node failures), local overload conditions occurring in the network, and security attacks of various sorts, as discussed below.

Work has focused on algorithmic aspects of the distributed aggregation of local measurements across the network, to produce global estimates. Specifically, work has focused on two main classes of algorithms, tree-based and gossip-based algorithms, addressing the following objectives: comparison of the two classes, its adaptation to various aggregation tasks, and evaluation with respect to the above comparison parameters. Our findings are primarily based on the following works:

- Designs and evaluations of tree-based and gossip-based protocols that are resilient to random node failures, either fully or partially [Wuh09a].
- Designs and evaluations of tree-based and gossip-based protocols for distributed threshold detection [Wuh08][Wuh09a].

In this section and partly in Section 8.3 we summarize our findings.

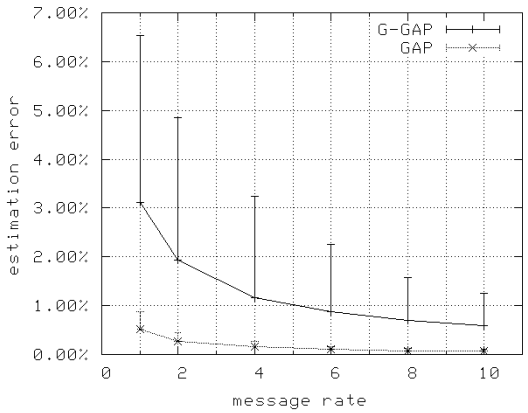


Figure 6-1: Estimation error vs. protocol overhead, from [Wuh09a]

6.2.1 Comparison of Gossip-based vs. Tree-based Algorithms, in Terms of Robustness and Performance

Overall, our theoretical and experimental investigations indicate that in a number of respects tree-based protocols are superior to gossip-based aggregation protocols we have been looking at, in terms of:

1. Overhead/accuracy ratios

2. Ability to recover from random failures

3. Ability to converge to accurate values in stable configurations. This is important for instance when cryptographic information needs to be aggregated, as is needed in e.g. privacy applications

4. Ability to support a wide range of aggregation functions

5. Fast convergence

6. Ease of analysis

For instance, in [Wuh09a] we consistently found close to a factor of 10 in reduction of the parameters we estimated while passing from a gossip-based to a tree-based solution. For instance we plot in Figure 6-1 the estimation error in terms of message rate for the robust gossiping protocol GGAP vs. the tree-based aggregation GAP developed in earlier work, using a trace based on real network data [Wuh09a]. Other experiments, e.g. measuring estimation error in terms of network size, or failure rate, or protocol overhead, provide results that are consistent with this observation. This applies also when comparing our gossip-based and tree-based solutions for distributed threshold detection.

For the other points in the list 1.-6. above we note first that there are standard, easy solutions available to make tree-based protocols robust against random node failures, at the cost of significant transient errors (cf. [Dam05]). We found that the problem of making gossip protocols robust is far more challenging. Indeed, we are not aware of a robust version of gossiping with a comparable scope as, e.g. the basic GAP protocol. In fact, the solution we propose in [Wuh09b] is robust only to random failures that are non-contiguous in the sense that no two neighbouring nodes of some given node is allowed to fail within a single round.

For the remaining points in the list 1.-6. we note that in general these are well-known standard points in favour of tree-based algorithms.

For a fair evaluation we also note that there are significant arguments against tree-based approaches. This includes the propensity to concentrate management traffic to a few

network links, and the - sometimes dramatic - transient errors which may arise as a result of tree reconfiguration in response to node failures. Also, we have encountered some indications that, at very high levels of churn, the performance of tree-based protocols begin to deteriorate so significantly that they begin to be outperformed by gossip-based ones. This point, however, is not yet understood well enough to make firm conclusions.

For the above reasons our conclusion is that, at this time, tree-based approaches are preferable to gossip-based ones for network management applications, under reasonable assumptions on node failure rates.

6.2.2 Aggregation Under Churn

The currently predominant protocol analysis techniques in the area of aggregation protocols focus on static network configurations, i.e. properties such as self-stabilization and convergence. After an initial period of instability, the network under examination remains stable from a certain point in time onwards. Protocol properties in unstable configurations are usually left to be examined by simulation or testing only. Unstable configurations are, for example, situations where state variables continually change, or nodes may fail and dynamically join the network. On the other hand, important protocol properties such as transient behaviour and accuracy cannot be fully understood without considering dynamic behaviour. For this reason, an important theme of the work has been the development and evaluation of stochastic models of tree-based aggregation under churn, i.e. where network nodes may be dynamically removed from, or join, the network.

Essentially, the idea is to build continuous time Markov models that capture some important system parameters, as they change during system execution, assuming a given rate of churn. Churn is measured as the rate of failure on a per-node basis, assuming that the size of the network remains constant over time (i.e., a roughly equal rate of node join and node failure).

We have developed several models. The models estimate network properties at varying levels of detail such as the number of nodes at a given tree level, or the aggregate held by a node at a given level in expectation. The models have been validated by simulation for varying churn rates and protocol rates.

6.3 Functionality

6.3.1 Threshold Crossing Detection

An important aggregation task is to detect network wide threshold crossings, for instance to determine that some network wide quantity such as average or peak load has been crossed, indicating a problem that may need attention. We have developed solutions for threshold detection based on both tree-based and gossip-based underlying protocols [Wuh08],[Wuh09a]. The desired functionality is illustrated in Figure 6-2.

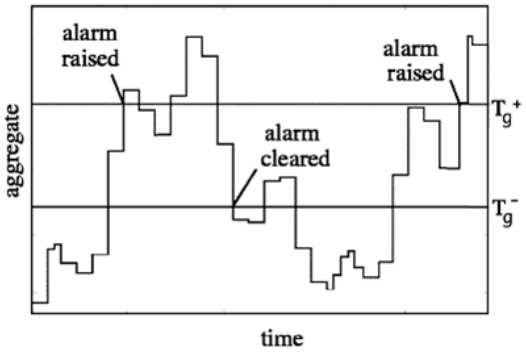


Figure 6-2: Desired threshold crossing detection functionality

Threshold detection determines an upper and a lower threshold with a hysteresis-like functionality. Crossing the upper threshold Tg+ from below causes an alarm to be raised, and crossing a lower threshold Tg- from above causes the alarm to be cleared again. This symmetry suggests a "dual mode" operation of the protocols which allows to "switch sign" of the detection machinery depending on which threshold has been most recently crossed. Exploiting this idea allows to define protocols in both tree-based and gossip-based variants that allow the overhead to almost completely eliminated for aggregates values that are far from the thresholds. This is shown on the simulation traces of Figure 6-3 for the case of the tree-based protocol. Similar results are obtained for the gossip-based version. In many applications where the thresholds indicate exceptional network conditions that are not expected to frequently arise, this is a very attractive feature, as this means that, under normal operating conditions, overhead from threshold detection can be neglected.

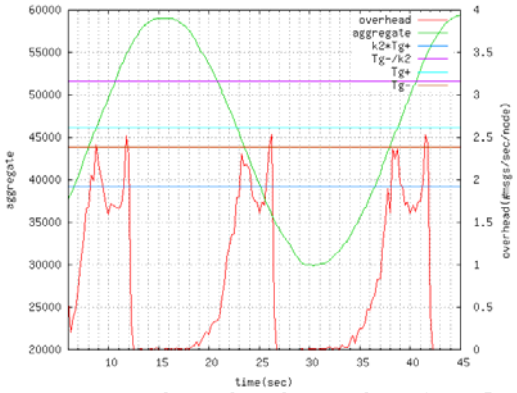


Figure 6-3: Protocol overhead over time, from [Wuh08]

6.3.2 Adaptive Avoidance of Network Implosion

"Not All aT Once!" (NATO!) addresses a problem that arises in many modern networks, such as sensor networks, grid networks, satellite networks, and broadband access wireless networks. Such networks consist of thousands of end devices (nodes) that are controlled and managed by a single gateway. At times, due to a state change or a local event, a large group of end nodes must send feedback messages to the gateway.

NATO! is a statistical probability scheme for precisely estimating the size of a group of nodes affected by the same event without explicit notification from each node, thereby avoiding feedback implosion. The main idea is that after the event takes place, every affected node waits a random amount of time taken from a predefined distribution, before sending a

report message (RPRT). When the gateway receives sufficient RPRTs to estimate the number of affected nodes with good precision, it broadcasts a STOP message, notifying the nodes that have not reported yet, not to send their RPRTs.

The gateway then analyzes the transmission time of the received RPRTs, defines a likelihood function, and uses the Newton-Raphson method to find the number of affected nodes for which the likelihood function is maximized.

Using mathematical analysis, we provide tight upper and lower bounds for the estimation error. We show that this error is approximately 1/(N - 1), where N is the number of sent RPRTs, and it is always over-estimated. We use this property to bring the estimation error very close to 0. Simulation results show that with only 20 feedback messages (coming from a group of 100 or 10,000 affected nodes), the estimation error is about 5 percent, and after error correction, the error is eliminated.

NATO! was presented in IEEE Infocom 2009 mini conference [Coh09].

NATO! Applications

NATO! is applicable for networks and systems that meet the following requirements:

- (R1) The network consists of a large group of end nodes reporting to a single gateway. Having each affected node sent a separate RPRT message to the gateway would result in one of the following implosion effects: (a) insufficient network resources for forwarding the messages to the gateway; (b) insufficient gateway CPU resources for processing all these messages; (c) delayed gateway response to the event.
- (R2) RPRTs are identical (e.g. Ack, Nack); If this is not the case, the gateway should form a Broadcast START message with a query that is expecting such RPRTs. NATO! Is not useful when RPRTs contain data that is unique to each sender.
- (R3) In order to correctly respond to the event, the gateway needs a good estimate of the number of nodes that have experienced this event.
- (R4) The estimation should be completed in a timely manner.
- (R5) The gateway is able to broadcast a STOP message to all of the nodes that might be affected by the event, to stop further transmission of their RPRTs.
- (R6) The setup allows for precise timing, i.e., (a) the event occurs at the same time, or the server can start NATO! by means of a START broadcast message; (b) all nodes are time-synchronized, or the network delays are known.

INM Applications for NATO!

As long as these six requirements are met, there are quite a few INM applications for NATO!. We describe two example applications:

Network Monitoring

Monitoring involves the collection of state and counter values from a large number of nodes at every time interval. This task utilizes significant network and CPU resources, which is generally addressed by aggregation techniques [Bra02], [Hei99], [Int00], [Sri01]. NATO! can be an interesting alternative to aggregation techniques.

NATO! is implicitly started at every agreed time interval. The managed nodes stay quiet, as long as its state is healthy, and its counters don't exceed an agreed threshold. The managed nodes send its RPRTs with bad news only. At every time interval, the gateway can precisely figure the severity of the problem by estimating the number of nodes that are in bad state. In order to adapt to new network conditions, the gateway can possibly broadcast a request to report on new threshold values only, or to change the time interval. In order to

localize the problem, NATO! can exploit the distributed manner of the INM environment, by implementing the scheme at each segment by a local gateway. Finally, corrective actions can be taken, once the severity and the location of the problem are identified.

QoS Management

QoS management can take advantage of the NATO! scheme, as described in Section 5.2.1. A cross-layer QoS MC needs to collect information from all nodes within its domain, including available transfer rate, one-way delay, BER, etc. However, this is not always possible, because not all nodes are integrating sophisticated mechanisms that are required to respond to such request. The NATO! scheme can implement an alternative approach; rather than collecting row information for QoS assignment, the QoS MC broadcasts a specific query, e.g. which nodes can accommodate a flow with specific QoS requirements. Only nodes that are capable of responding to such query, and can accommodate such request, start NATO! (for a delayed positive response). The NATO! gateway algorithm precisely estimates the number of nodes that can accommodate such request, without explicit notification from each one. With this information, the QoS MC can determine if QoS-aware routing is feasible for a given request.

6.3.3 Topology Discovery

Network bootstrapping and discovery are two essential mechanisms to ensure the proper information dissemination in distributed MCs. Bootstrapping corresponds to the initial warm-up of the network (or new entity) where static properties are learned by each SE (e.g. policies, capabilities or topology). Discovery refers to the continuous process of maintaining the information updated (including network status). These mechanisms become more complex and less efficient with the increasing number of nodes in the network. We propose *Hide and Seek (H&S)* [Gua09], a new algorithm for network discovery, and information propagation and synchronization. H&S is essential to ensure the existence of relevant and sufficient information on each SE for self-adaptive decision-making processes.

Two roles were considered in H&S: seeker and hider. A *seeker* node sends directional contact messages to its neighbourhood using a *probabilistic eyesight direction* (PED) function. This function aims to narrow the directions through which contact messages are sent. Once contacted, *hider* and seeker synchronize their knowledge, keeping track of each other. The *hider* becomes a new *seeker* and the process is repeated until all nodes have been contacted. Using a probabilistic directional search provides better scalability support and improves efficiency of the search mechanism when compared with other approaches, such as gossip (probabilistic flooding) and non-controlled flooding.

To evaluate the performance of H&S we have implemented a simulation scenario in MATLAB where several static nodes were randomly distributed in a specific area (Figures 6-3 and 6-4). Depth first search (DFS) gossip flooding and non-controlled flooding algorithms provided additional results for comparison purposes. The metrics considered were the used simulation cycles and the number of messages exchanged. For both, H&S revealed the best results (Figure 6-5).

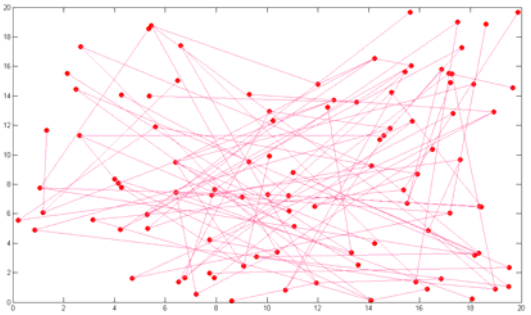


Figure 6-3: H&S algorithm

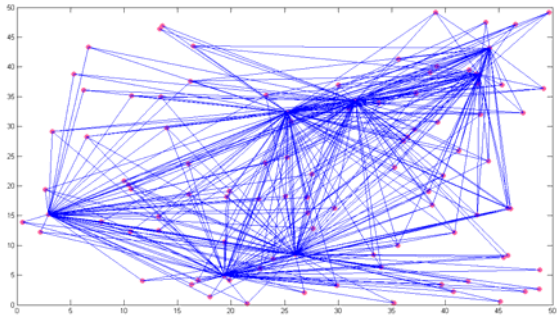
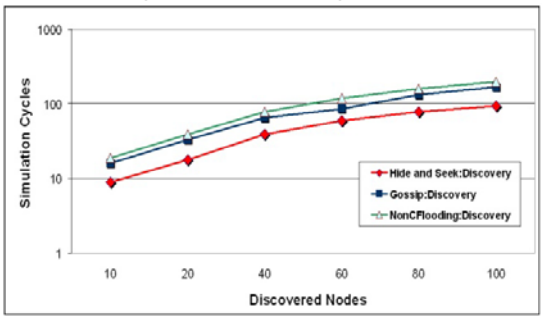


Figure 6-4: DFS-Gossip flooding



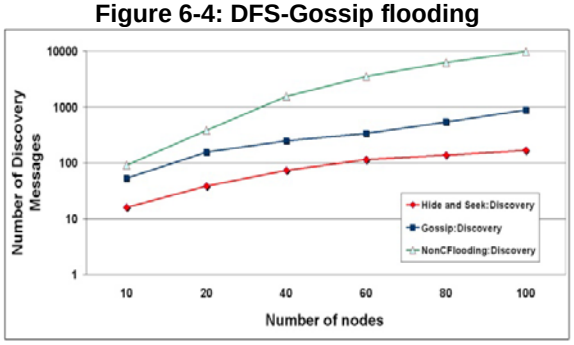


Figure 6-5: Comparison between H&S, DFS-Gossip and Non-controlled flooding approaches

The results analysis allows concluding that H&S records lower messages overhead for information synchronization. The algorithm's complexity shows to be linear O(n-1). The other two have a quadratic O(n(n-1)) complexity.

Conclusions

H&S was applied to a very simple scenario to evaluate its behaviour when compare with other approaches. It presents good scalability results since it requires fewer messages and has a linear complexity.

We plan to evaluate and extend the usage of H&S:

- Evaluating its behaviour when dealing with partial information in the context of task 4.1 (considering that only part of the information can be conveyed in each message).
- Studying its application to VNet scenarios in the context of task TC34 (request dissemination, network mapping and VNet recovery).

6.3.4 Search in Dynamic and Self-organizing Networks

Search and routing in dynamic, self-organizing networks usually cannot rely on stable topology from which search tables, shortest paths and other optimized access techniques are derived. When no reliable indices or routing tables are available, flooding, random walks or gossip-based methods have to be considered to explore the network. These approaches can exploit partial knowledge available on the network nodes to reach a destination, but the search effort naturally increases with the lack of precise information due to network dynamics. This problem is especially relevant for wireless technology with strict limitation on power consumption. We address the efficiency of random walks and flooding for exploring networks based on case studies evaluated by simulation and transient analysis. In this way, performance tradeoffs are demonstrated when combining shortest path routing with

randomized techniques. There are at least three networking scenarios which lead to increasing dynamics when integrated into future Internet structures

- Peer-to-peer (P2P) and other self-organizing overlays on the IP infrastructure,
- mobile ad hoc networks (MANET) and
- sensor networks.

A search may refer to users, network nodes, information, content or services of any kind residing on network resources based on unique identifiers like IP addresses or hash values as often used in P2P networks. Developments on top of P2P systems have advanced towards distributed databases systems, which can be built in a scalable and efficient way even on unstructured network topologies and in the presence of unreliable nodes.

Therefore the Bubblestorm approach [Ter07] set up a randomized replication scheme, where a data item is distributed on a set of nodes forming a data bubble. The size of each data bubble is kept proportional to the square root of the network size, which can be effectively estimated by the NATO algorithm or similar techniques discussed in Section 6.3.2. The replication scheme achieves a high reliability and improves the performance of search as well as the throughput by enabling multi source downloads. Experience of the Bubblestorm architecture in sufficiently large networks has shown that a simultaneous disappearance of up to 90% of the nodes e.g. caused by breakdowns in the underlying transport network still leaves the remaining network and database system intact.

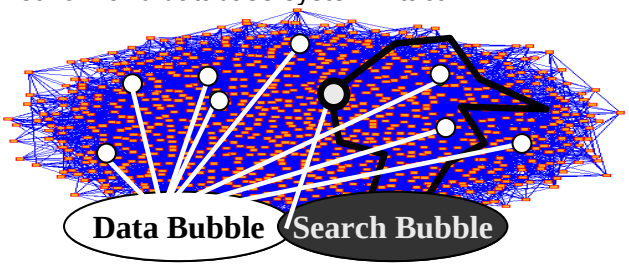


Figure 6-6: Bubblestorm. Random search for replicated data in self-organizing networks

Comparison of search performance for random walks and flooding

The search can be done by flooding a request or by random walks in order to exploit the self-organizing network. In order to reduce the messaging overhead, flooding is usually restricted by a predefined hop limit h, which is set with regard to knowledge of the network structure and demands for coverage. Alternatively, a small initial value for h may be stepwise increased if the search radius turns out to be insufficient. But then a new step revisits all the nodes of the previous step and as another disadvantage, the number of nodes being reached for a larger search radius is not known a priory and is often increasing exponentially in unstructured or scale-free networks.

Random walks

Randomized techniques approve to be useful in the construction and exploration of self-organizing networks, when the maintenance of search tables and structured indices is expensive. There is considerable work in recent time on how randomness can help to manage dynamic networks with minimum overhead while preserving sufficient connectivity. A basic random walk exploits the network as a stepwise process, which proceeds from a node to a neighbor at the next hop.

A random walk R of length L is denoted by the series $r_0, r_1, r_2, \ldots, r_L$ of visited nodes, where an edge (r_{k-1}, r_k) in the topology is chosen randomly for the k-th hop ($1 \le k \le L$). Usually a random walk chooses its next hop with the same probability among all neighbors of the currently visited node: $\forall\ a, k;\ (a, k) \in E$: $p_{ak} = \Pr(r_{n+1} = k \mid r_n = a) = 1 / d(a)$, where $d(a)$ is the number of neighbors.

The corresponding transition matrix $P = (p_{ak})$ determines a random walk as a Markov process, where the network nodes directly correspond to the states of the underlying Markov chain and edges to allowable transitions from one state to another. Thus the behavior and performance of random walks can be evaluated using transient analysis [Hass09] in addition to usual simulation studies.

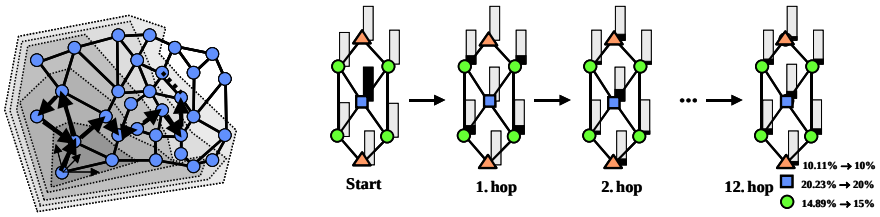


Figure 6-7: Flooding (gray surfaces indicating distances) versus random walks (arrows) **Transient behaviour of a random walk (bars represent the sojourn probabilities at the *m*-th hop)**

The main drawback of a single random walk is the long delay while it may take some winding route through the network. Flooding spreads search messages in parallel to all nodes in the neighborhood up to some hop distance *d*. In graphs with small diameter it is difficult to find an appropriate search radius *d* to cover a predefined number of network nodes, which contributes to the disadvantage of a large messaging overhead. This motivates to propose new routing schemes for sensor networks or to combine random walks with flooding.

Combined variants include

- random walks with an additional flooding step with small radius from all nodes being traversed or from the last node,
- starting several random walks in parallel or branching a random walk into multiple paths.

As a rigorous overhead control scheme, the number of messages in a random walk, flooding or combined search can be limited by a time to live counter, also denoted as *budget controlled search*. When the search is split up in multiple paths being traversed in parallel, the budget must also be split.

A randomized replication of data as in the Bubblestorm database architecture provides a favorable environment for random walk searches. Besides studies showing favourable properties of random walks in large unstructured networks, other investigations proposed random schemes for ad hoc and sensor networks. In addition, it is known that random walks can efficiently exploit imprecise and only partially valid information in support of a search or when many nodes in the network are able to respond, i.e. when it is sufficient to reach one node in a larger set.

Multiple random walks in parallel

Random walks often can reduce the communication overhead, but they traverse the hops sequentially and thus usually spend much more time than flooding. Multiple random walks in parallel may be applied in a compromise between demands for low delay and low overhead. If a random walk is assured a success rate of $\sigma < 1$ within *m* steps, then *k* random walks of the same type in parallel each with m steps reduce the failure rate from $1 - \sigma$ to $(1 - \sigma)^k$. Thus a success rate of $\sigma = 90\%$ is improving up to 99.999% when 5 random walkers are combined in parallel. A single walk often achieves this success rate in less than *mk* steps but even then needs up to *k*-fold time.

Biased random walks using partial routing information

When we assume that nodes can partly use valid routing information to forward a search request or otherwise forward it to a randomly chosen neighbour, then this leads to biased

random walks combining deterministic and random steps. The mean search time of biased random walks usually grows linear with some factor times the hop distance to the destination, where the factor is roughly proportional to the ratio of random to deterministic steps. A bound and case studies for the favourable behaviour are given in [Hass09].

For larger networks, self-organization is often combined with a hierarchical structure, e.g. by super nodes in eDonkey and other P2P networks or by backbone nodes which subdivide wireless or sensor networks into different areas assigned to them. Thus hierarchical structuring is a means to limit the size of self-organizing network areas with decisive influence on the search and routing performance.

6.3.5 Anomaly Detection

We have developed a distributed approach to adaptive anomaly detection and collaborative fault-localisation. The statistical method used is based on parameter estimation of Gamma distributions obtained by measuring probe response delays. The idea is to learn the expected probe response delay and packet drop rate of each connection in each node, and use it for parameter adaptation such that the manual configuration effort is minimised. Instead of specifying algorithm parameters in time intervals and specific thresholds for when traffic deviations should be considered as anomalous, manual parameter settings are here specified either as a cost or as a fraction of the expected probe response delay (see [D4.2] and [Ste09a]).

The approach to distributed anomaly detection has been extended to take into account detection of deviating probe response delays which is achieved by learning temporally overlapping statistical models [Ste09b]. This type of learning mechanism includes temporally palimpsest properties and allows for smooth adaption to long-term changes, while gradually forgetting earlier observations. Initial performance tests of the extended anomaly detection algorithm show that link and node anomalies caused by either deviances in expected probe response delay or communication failures can be detected and localised to a certain link or node with satisfactory performance.

Statistical model and parameter estimation. During run-time, observations of probe response delays obtained by probing are continuously collected, forming a two-parameter Gamma distribution. The choice of model is motivated by the assumption that the response delay is a sum of independent exponential transmission delays caused by e.g. queueing times in processing nodes. Similar assumptions about network traffic delays (on different network levels) matching Gamma, Weibull, or other exponential distributions have been made in a number of papers, e.g. [Fär02], [Cho99]. The parameters of the distribution are estimated via a method of moments approach based on the first and second sample moments (e.g. [Kum06, Hua06]). The probing intervals are autonomously adapted based on the expected probe response delay and a cost. The cost is set manually and represents the trade-off between detection performance and probing traffic induced by the algorithm. Here, two types of intervals are used such that link load caused by probing traffic is reduced during normal network behaviour, while being somewhat increased when a potential anomaly is about to be detected. Compared to ordinary monitoring with fixed probing intervals, the use of two probing intervals that adapt to current network conditions can reduce the total link load caused by probing. Experiments performed in the OMNET++ simulator environment also show that the traffic generated by the anomaly detection algorithm scales linearly with the number of network connections (Figure 6-8). In the experiments, scale-free networks of increasing size were generated via the Barabási-Albert method [New03].

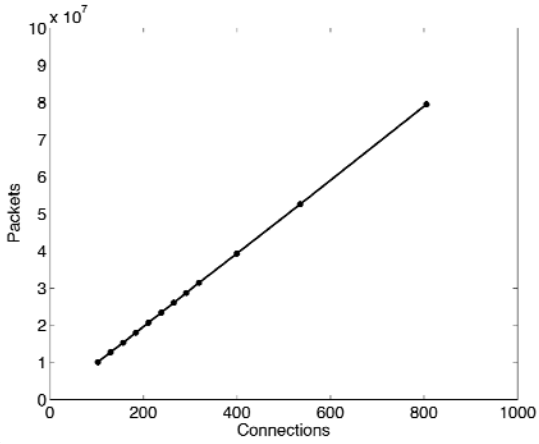


Figure 6-8: Scalability tests for different sizes of scale-free networks.

To account for long-term variations in the network, each node models probe response delays using overlapping Gamma distribution models, using the previous model as prior to the next model. The learning scheme is circular with $M = N/T$ models, each based on N observations and the degree of 'forgetfulness' T (Figure 6-9).

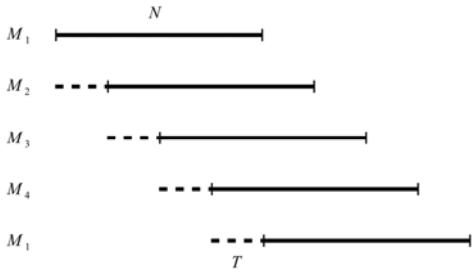


Figure 6-9: Overlapping parameter estimation.

The degree of overlap T controls the temporally palimpsest properties (i.e. forgetting models over time). By using previous model as prior input to the next model, a smooth transition between models is achieved while previous probe response delay observations have smaller impact on the current parameter estimations. The benefit that this learning scheme offers is faster adaptation to new network 'regimes', caused by e.g. software upgrades and change of network equipment.

Detection models and localisation of anomalies. In the following model, we assume that the probability of receiving a probe response is

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

where $P(D)$ is the measured packet drop rate on the link and $P(t)$ is the Gamma distribution of observed probe response delays. Using this model, anomalies related to communication failures on either links or nodes are detected using the following decision model:

$$P(\neg R | R_{\Delta t}^{(1)}, R_{\Delta t}^{(2)}, \cdots, R_{\Delta t}^{(i)}) = \prod_i 1 - P(R_{\Delta t}^{(i)}) < \psi \quad (2)$$

Assuming independency between probes, an anomaly has been detected if a series of probes are sent without any reply on a connection. In case of a detected anomaly a fault-localisation process based on node collaboration is initiated to pinpoint the origin of the anomaly to a link or node. The number of probes needed to detect an anomaly is adapted based on both the expected probe response delay and the packet drop rate measured on the link. Hence, with a sufficiently small value on the detection threshold ψ, robust anomaly detection with increased confidence and few false positives can be achieved [Ste09a]. An example of the adaptive behaviour is shown in Figure 6-10, in which the detection performance was tested for increasing packet drop rates. In the experiments anomalous events (communication failures and probe response delay variations) on both links and nodes were simulated in OMNET++ [Ste09b]. A scale-free network of 30 nodes and 81 undirected links was used as input to the simulator.

In order to detect deviating probe response delays, the current model M_{i+1} is compared to the previous model M_i using the symmetric Kullback-Leibler divergence for Gamma distributions as a metric [Kwi08]. Changes in the observed probe response delay on the link are detected when the divergence $KL(M_{i+1}\, M_i)$ is higher than the threshold η [Ste09b]:

$$KL(M_{i+1}M_i) = D(M_{i+1}||M_i) + D(M_i||M_{i+1}) > \eta \quad (3)$$

The threshold controls the detection sensitivity, and can be set to detect very small changes in the probe response delay of a link occurring over a longer period, or extreme changes over shorter time periods. The detection of a probe response delay deviation on a link will trigger an alarm and the neighbouring node will get notified as well. In case all links between a node and its neighbours have detected probe response delay deviations, the node sends an alarm about the anomalous node behaviour. An example (using the same experimental settings as described above) of the detection performance is shown in Figure 6-11.

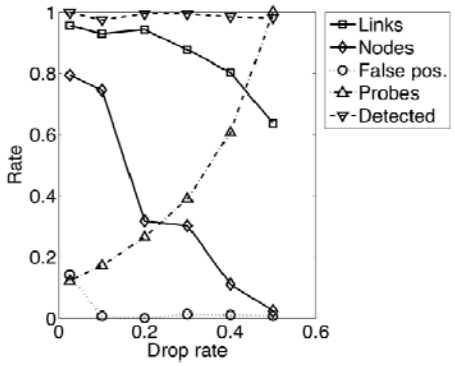


Figure 6-10: Performance of detection and localisation of communication anomalies on links and nodes.

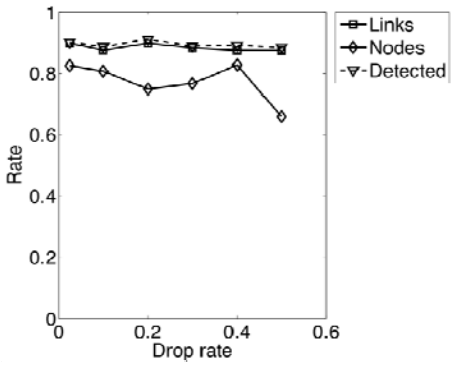


Figure 6-11: Performance of detection and localization of probe response delay variations on links and nodes.

Further improvements. Detection of probe response delay variations generates a burst of alarms until the learning model has collected enough observations to reach a converged state. To avoid bursts of alarms, the algorithm currently uses simple thresholds to avoid sending more than one alarm per detected deviation. One way to address this problem is to develop improved alarm-filtering mechanisms using e.g. Poisson distributions for individual network equipment. Further, we believe that a similar approach can be used to detect abnormal behaviour for small populations of links and nodes in local regions of the network.

6.3.6 Secure Aggregation

The aggregation protocols used in the context of INM involve large numbers of network nodes (routers) collaborating to produce performance or security related statistics on huge and generally incomplete networks. The aggregation process often involves security or business-critical information, which network providers are generally unwilling to share without strong privacy protection. This appears to be an essential obstacle to enable the INM approach to be deployed in multi-domain, cross-provider settings.

It is therefore important to develop versions of the INM algorithms which are able to execute without providers private information leaking to outsiders. This is particularly important for network management information, as this generally contains lots of information about the configuration, operation, load, and performance, of the providers' internal network. In today's SNMP-based management architecture, protecting this information can be done by point to point encryption and authentication of the network management traffic. In an INM-based solution this is, however, much more complicated, since the INM protocols are based on information exchange between large classes of nodes, including nodes that link, e.g., competing provider domains. Hence, end-to-end encryption and authentication is no longer sufficient to protect the privacy of internal provider domains, and a different type of solution must be sought.

One possible approach is to use techniques known from secure multi-party computation (SMC), where the objective is to compute, in a secure and privacy-preserving fashion, an arbitrary computable function, distributed among a small number of fully connected agents. The difficulty is that the INM network calls for very large network graphs, whereas SMC works for graphs with a small number of nodes only. A typical attacker can be thought of as a provider wishing to monitor the traffic densities, size, and configuration of a competitors' internal network. We have so far examined the case of passive, "honest-but-curious" attackers. In this scenario it is assumed that it is sufficient to protect only against passive attacks, where the attacking agent can be counted on to correctly execute the joint INM protocol, but who will want to apply any data mining techniques available to extract information from the information it legally has received.

The key to solve this problem is to design algorithms in which cryptographic, or information hiding techniques are used to hide the "semantics" of any information exchanged between nodes as the protocol executes, and only allow this semantics to be extracted at the aggregate level. The algorithms are subject to the same design requirements as the other INM protocols we have developed, regarding, e.g., scalability and efficiency. In fact, due to the use of information hiding techniques/cryptography, the requirements are in effect stronger, as the aggregates must be computed exactly, in order for the aggregate information to be computed correctly. For this reason attention has so far focused on tree-based aggregation, as gossip-based approaches are known to only converge asymptotically.

So far, the standard aggregation functions sum, average, min, and max have been considered [Kre09]. For sum and average, we give a protocol that is information theoretically secure against a passive adversary, and which requires only one additional round compared to non-private protocols for computing sums. The solution is easy, and based on each node computing a pseudo-random integer which is used as a random seed in the first round of neighbour interactions. Min and max are more difficult. Our investigation has started with disjunction, logical "or", as this is corresponds to max in a 1-bit data domain ordered in the normal way. The task then is to compute the "or" of n bits in such a way that at least one bit is high if and only if the output bit is high, but it should not be possible for any participant in the protocol to say anything about other participants except that, if the "own" bit is 0 and the output is high, then some other participant has a high bit. We present two solutions to the "or" problem, and show how this can be used to solve the general min/max problem, essentially by computing the min/max bit for bit in a fairly obvious way. The first solution is computationally secure, and the second is information-theoretically secure. The latter uses a general composition approach which executes the sum protocol together with a standard multi-party

protocol for a complete sub-graph of “trusted servers”. This can be used, for instance, when a large network can be partitioned into a smaller number of provider domains.

6.3.7 Aggregation for Reputation Systems

Valuations systems may be seen as a generalization of the quality of service approach in networks. By knowing the value of given resources and actions for given actors, smarter decisions can be taken to achieve the objectives. In 4WARD, we show how a valuation system excels over conventional Intrusion Prevention Systems (IPS) in handling overload and attacks to networks.

Reputation/Valuation Model

Whether we speak of establishing reputation or establishing a valuation, the objective of a valuation system is always to express the value of another actor, service or resource for a specific self with a specific objective. From this definition alone, we can extract four key concepts:

- A valuation expresses benefit
- A valuation is subjective: It expresses the expected benefit for a specific actor.
- A valuation is specific to a given other actor or action.
- A valuation is contextual: The value of a given resource will not be the same, depending on what we are aiming to achieve.

We will handle these constraints with the following model: We define v(id1, id2, c) an unit-less value in the range [-1;1], which expresses the Value/Cost ratio of id2 (a given resource or actor) to id1 (the valuating actor) in the context c. This value shall semantically express a relative, subjective benefit of the client associated with the valuated identity to the service. Note that, since we will be exploring the concept in closed scenarios, we do not explicitly represent the context of the valuation in the model.

The model and the validation framework which has been developed in the 4WARD project allows for a generic transfer of application level knowledge to in-network management functions. In the following, we will inspect the case of network overload by denial of service attacks or legitimate usage.

Application to Network Overload

Services in a network are protected by the Distributed, Context-Aware Firewall (D-CAF), developed in the 4WARD project. The firewall resides at a bottleneck in the topology, i.e. the possible incoming traffic volume can be higher than the downstream network links. The firewall is able to measure traffic volumes, on a per-flow level, and to filter traffic. It is also able to receive valuation reports from the services, which assign a subjective value to any given identity (represented here by IP addresses or IP ranges).

Typical overload situations in such a scenario are either illegitimate denial-of-service attacks, or legitimate flash crowds. In both cases, the sudden surge of traffic exceeds the available resources in the network. The system will observe that a given traffic threshold has been passed and begin to use the collected reports to filter. The rationale is that, whether a resource consumption attack takes place or not - if the services can handle it, there is no need to filter possibly legitimate traffic and endanger the business case. However, as soon as there is overload and the system in danger, a filtering decision has to be made. Here, the valuation reports coming from the services will be exploited. In contrast to conventional Intrusion Prevention Systems, there is no heavy burden resulted from analyzing the traffic for benign and malign behaviour patterns: The applications, which have to perform this task anyway, can

report their findings in a cross-layer, collaborative manner to the network components. The system is described in detail in [Var09].

Aggregation and Re-Distribution

Since the valuation reports are always of a subjective nature, and several services may report conflicting values for the same identity, aggregation of the valuation data has to take place in the management component.

The advantage of the semantic simplicity of the approach is that any aggregation of valuations in the firewall has the same meaning as each individual reports: It represents a subjective view of the relative benefit of identities to a given service. Hence, aggregated valuation reports from one D-CAF instance (or in-network management components in general) can be re-exported to other instances in the same administrative domain, or even across administrative domains. Both vertical and horizontal distributions can be envisioned: If valuations are re-distributed towards the next larger firewall (situated even closer to the core of the network), the valuations from several protected networks can be aggregated to be used in the case of a more severe overload. In a horizontal distribution, D-CAFs of several networks may exchange their relative subjective valuations to get a better insight on how users behaved in the neighbouring domains.

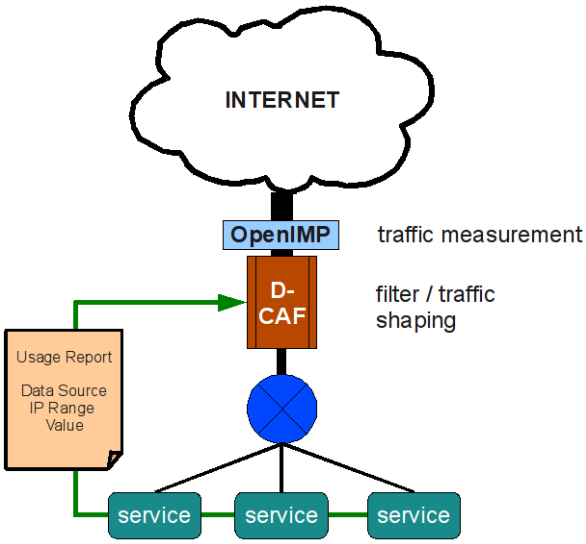


Figure 6-12: Using Valuation for Mitigating Network Overload

6.4 Conclusions

In this chapter, we have presented our work on situation awareness, which is responsible for estimating the network state and is the necessary input to self-adaptation mechanisms. We have focused on a subset of the management tasks involved in situation awareness. Specifically, we have worked on real-time monitoring of network-wide metrics, group size estimation, topology discovery, data search and anomaly detection. The chapter discusses algorithms aspects of distributed network monitoring and presents a number of algorithms providing key functionality for the management of the future Internet.

7 INM Self-adaptation

7.1 Introduction

Chapter 7 discusses network self-adaptation in the scope of INM. The network itself takes proactive or reactive network management actions for the purpose of recovering a fault, avoiding a predicted fault, optimizing the network operation, or enforcing new or modified objectives submitted by the network operator, by changing the network configuration, the network setup, or resource allocation.

Chapter 5 presented the INM framework with precise structural and functional definitions. Chapter 6 discussed situation awareness, which is the basis for learning the current state of the network and its operation. This chapter makes use of this knowledge in order to take corrective actions.

The discussion about self-adaptation is arranged as follows:

Section 7.2 investigates the characteristics of self-adaptation in the context of INM capabilities.

Sections 7.3 through 7.8 are each looking into one aspect of self-adaptation.

Adaptive self-organization of the management plane is studied in Section 7.3. An abstraction layer is proposed, in which synchronous events are aggregated, in order to facilitate distributed INM and scalability.

In Section 7.4 we show how to design network protocols suitable for self-adapting INM operations, based on molecule-like entities such that the dynamics of execution flows in the network can by analyzed as if they were chemical processes.

INM-integrated configuration planning and verification functionalities are proposed in Section 7.5. This mechanism predicts the possible outcome of a configuration change request, thereby ensuring INM-built-in stability

Section 7.6 presents algorithms that adapt resource requests to the network capabilities, which are discussed in the context of QoS for Vnets, and further elaborated in Section 9.3.

Sections 7.7 and 7.8 discuss two self-adaptation concepts that are closely related to the management of GPs: a congestion control scheme which exhibits emergent behaviour, and self-adapted routing schemes for wireless multi-hop networks. The concepts are briefly presented here and are further elaborated in chapter 9.

Finally, Section 7.8 concludes the self-adaptation chapter.

7.2 The Features of Self-adaptation

In this section we investigate the characteristics of self-adaptation, namely, how the network is dynamically adapting to changing conditions (e.g., congested path, broken link, new node, new objectives submitted by the operator). Our view is that the network is self-adapting by means of self-adapting MCs (Section 5.2.1). MCs implement INM algorithms that should contain self-adaptation control loops. The scope of this section is to investigate how MCs are internally organized in order to facilitate self-adaptation.

The self-adaptation control loop is an algorithm or a portion of an algorithm that is subject to the same behaviour rules as any INM algorithm. More specifically, it publishes its capabilities, and it is part of the mapping process between high-level objectives and the INM algorithms that fulfil them.

There are a few possible design options for the self-adaptation control loop. As depicted in Figure 7-1, self-adaptation control can be fully embedded within the MC in a monolithic manner (picture a), or represent an identifiable entity inside the MC (picture b). One

can also envision an MC that is dedicated for the self-adaptation control loop (picture c), which enables self-adaptive behaviour of other INM MCs that are otherwise not self-adaptive (picture d). In all variants of the design space, MCs exchange data and self-adaptation control information, which are communicated across MCs using the collaboration or the organization interfaces (Section 5.2).

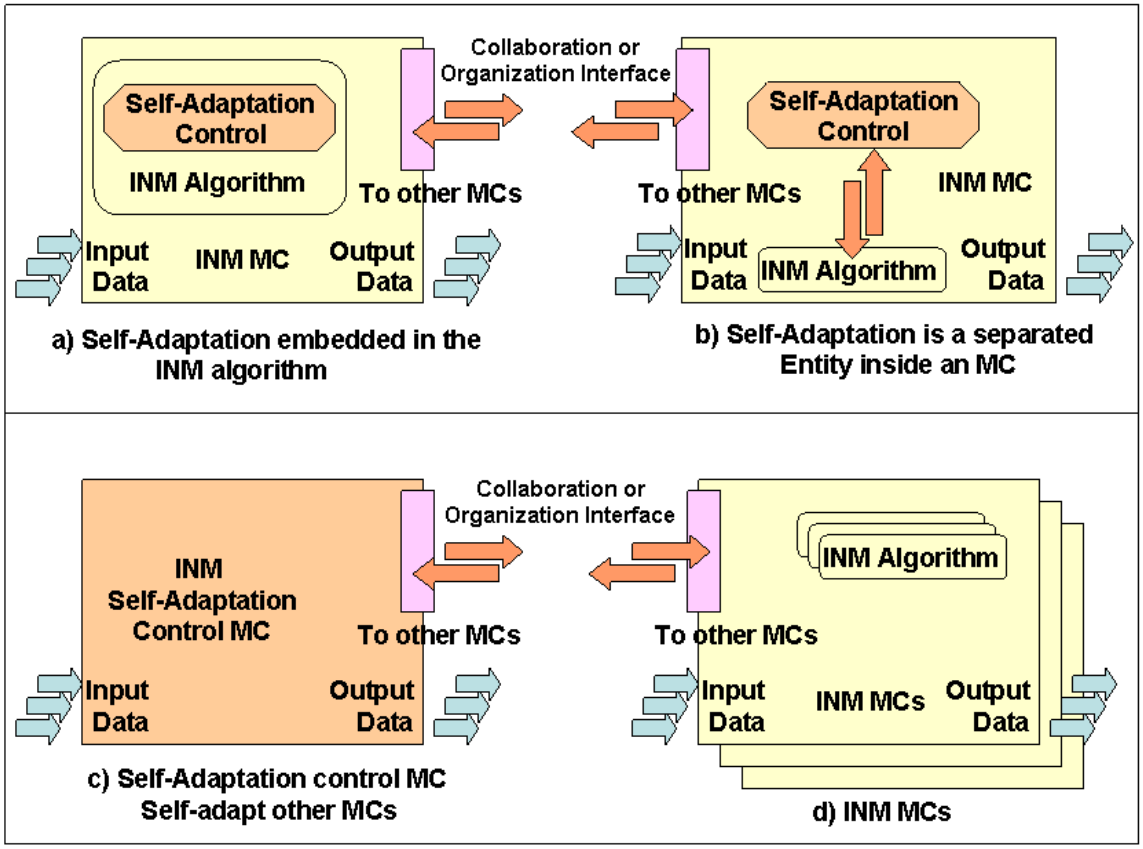


Figure 7-1: The design space of self-adaptation functionality within MCs

Another aspect of this analysis is whether the self-adapted MC resides in one SE, and thus functions somewhat independently, or when collaborating self-adapted MCs reside in multiple SEs, possibly in different network elements. The first case is rather simple, and the decision how to realize self-adaptation is an implementation issue that is specific to the algorithm (design options (a) or (b)). The second case represents a more general and interesting scenario, and the rest of the analysis focuses on appropriate design options for it.

Generally speaking, the embedded and separated designs, (a) and (b), seem to be more appropriate for distributed architectures, whereas design option (c) lends itself better for centralized implementation (or at least, less distributed). This is due to the fact that a distributed architecture necessitates the presence of self-adaptation logic in every MC, and self-adaptation is realized by means of collaboration between multiple MCs. Consequently, design options (a) and (b) are also more flexible and robust, due to the fact that any of the MCs can be the master for coordinating the self-adaptive process.

During our investigation, we have identified one neural network algorithm for QoS management [Alo07], which is not decomposable into smaller parts without loosing functionality. We refer to this type of algorithms as atomic. If an INM MC is realised by using such neural network algorithm, then self-adaptation cannot be separated from the rest of the

algorithm, and thus, only the embedded/monolithic design is valid (picture (a)), comprising an atomic INM algorithm. Having one such example makes us conclude that in any case where atomic INM algorithm is involved, the embedded/monolithic design is the only option.

It is our view that each one of the presented design options is valid. They all demonstrate the advantages of self-adaptation. Any MC that implements a non-atomic INM algorithm can use any of the suggested design options. The design option selection is really a design issue, taking into account the specific nature of the algorithm, performance and overhead considerations, as well as the architecture used (i.e., distributed vs. centralized). The various self-adaptation algorithms that are described in subsections of this chapter exemplify the validity of this conclusion: for each design option we have identified algorithms for which the design option is best.

The following are guidelines for selecting the most appropriate self-adaptation design option

- Design options (a) and (b) are a better fit for a fully distributed INM architecture, whereas design option (c) is more appropriate for centralized approach (or at least less distributed)
- For very large domains, a distributed INM architecture is the only scalable option, and thus, design options (a) and (b) are preferred
- When the INM algorithm is complex and requires significant amount of processing/storage resources, it might be more appropriate to locate it in a designated SE that is capable of hosting it, thereby preferring design option (c)
- When self-adaptation logic cannot be separated from the rest of the INM algorithm (an atomic INM algorithm), or when it is extremely difficult to separate them, then design option (a) is preferred
- When each SE act somewhat independently, relying only on input from its neighbours, design options (a) and (b) are preferred
- Design options (a) and (b) demand more networking resources than design option (c) (i.e. higher overhead), and are therefore less appropriate in situations where networking is scarce or more expensive

It is desired that the special case of atomic INM algorithms can be made general, such that any design can be selected. We therefore propose that an atomic INM algorithm is embedded into a more general self-adaptation logic, which can now be designed to use any of the design options presented. This concept is depicted in Figure 7-2, which extends the embedded/monolithic design option (picture (a) of Figure 7-1). The same concept can be applied to the separated and the dedicated design options (pictures (b) and (c) of Figure 7-1).

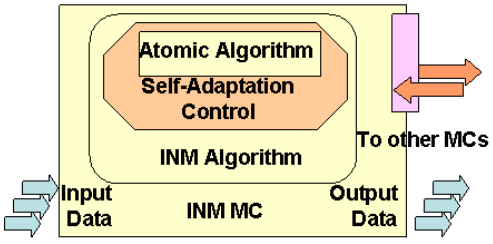


Figure 7-2: Atomic INM algorithm embedded within self-adaptation logic

7.3 Self-organization of the Management Plane

Self-organization is an important feature of the INM framework that builds a very simple interface for network operators to control the INM functions. Once the different algorithms are deployed in the network, it is required to organize their interfaces and objectives, so that interface exposed to operators hides much of the complexity of the underlying algorithms.

For this purpose, INM provides a clustering capability that provides self-organization with respect to the following two functionalities:

a) It builds an abstraction layer where objectives are composed towards the interface to the operator and disseminated from the operator's interface to the distributed INM algorithms. As a rule of thumb, 10 objectives should be presented to the operator.

b) It changes its composition scheme, so that the objectives reporting critical conditions in the network are reported to the operator's interface; in other words, objectives that were hidden before, can be taken out of the composed information and made visible for better optimization.

The clustering capability can be seen as the "glue" that composes different capabilities together into ordered management operations. Additionally, the capability is designed to take in consideration the trade-off existing in a distributed architecture: if we compose an information element with other ones, we are able to control more resources with few management operations, but on the other side we lose some level the efficiency in controlling those single resources. This trade-off is here taken under control, in the sense that the composition scheme is adapted to keep the lost of performances within business objectives.

On the upper level, the clustering capability is also the means of the construction of the operator's console. Figure 7-3 shows this in few sequential steps that represent the bootstrap sequence of an INM network, assuming all distributed INM capabilities are turned on at the same time. INM capabilities announce their objectives through asynchronous events. When the clustering capability is turned on, it starts to compose objectives in a step by step manner. At the end, only some high-level information is made visible, the list of composite objectives.

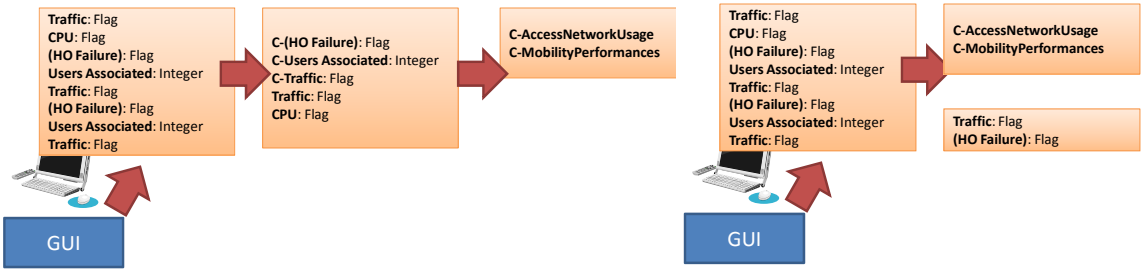


Figure 7-3: Creation of composed objective into operator's console.

The self-organization mechanism takes place also during normal operations –after the bootstrap– to adjust the composition rules of the clustering capability. The clustering capability is in fact listening for alarms, and isolates the objective that has been violated. This objective is then re-activated and isolated from the other composed objectives; its alarm can be handled independently. The alarm of the isolated objective is then treated following usual INM procedure: either (i) a capability can handle it and enforce an adaptive action or (ii) the alarm is reported to the operator as exception. The handling of exception is treated following the previous work [Min09].

For the first case, we consider the use case of management of a large scale radio access network, where it is necessary to configure a high number of base stations (order of

ten thousands for LTE deployment in a medium size country like Germany). In this case, operators rely on a limited set of configuration parameters, but these parameters are necessarily composite information, that is then mapped by the network in fine granular objectives on each capability. Also self-optimization functions (SON functions) can work through aggregated information, especially for mobility parameters. If this occurs, the effect of the SON function is partially inaccurate, because it is based on composed values. If a "hotspot" –that is a limited set of base stations with bad performances or with anomalous traffic – appears in the network, the SON function does not have the means to intervene on it with optimal fine-granular configuration. The clustering capability introduces instead self-organization behaviour exactly to support such fine granular optimization processes, while still maintaining scalable composition of objectives for non affected nodes.

In the second case, we consider the same console as presented above. In this case, part of the objectives is isolated and, since their alarms have not been handled, they are reported to the console for manual intervention. Figure 7-5 shows the visual effect on the console. The final effect is that operators can rely on self-organization as a reliable instrument to control large-scale networks. At the same time, they are guaranteed to still have fine control in case of anomaly situations.

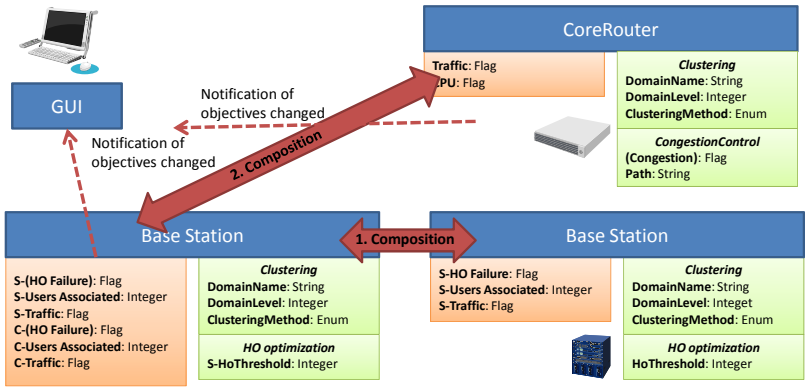


Figure 7-5: Composition steps between distributed capabilities.

The clustering is designed as a standalone capability that can be used by the other capabilities, both for monitoring as well as adaptation. A cluster is defined with two elements: (i) a cluster name is the name of the cluster. It is associated to an entity as a "tag"; its definition can follow the rules for the creation of domains, as explained in Section 5.2.3. (ii) a cluster level is a number that defines the authority level of different entities within a cluster. On a large scale network, this level allows an operator to define some specific order rules to execute composition and therefore makes it possible to introduce some authoritative decision points in INM.

7.4 Enabling Self-adaptive Processes in INM Using Chemical Network Protocol Design

The deterministic character of today's protocols is not needed everywhere in networking; in many network management services (routing, load balancing and long lasting signalling streams) we can relax the level of certainty for the purpose of obtaining more resilience and robustness collectively by the involved network elements. As is conjectured in [Sha02], softening the precision of the protocol's specification may avoid brittleness of the system by gaining resilience to various perturbations, such as the partial loss of information in form of packets or code, through a constantly adaptive operation that follows the network dynamics

[Mey08]. To this end, chemical networking employs chemistry as the central dogma for designing network protocols with transposed properties from chemical systems, such as continuous self-adaptation, which allow a system to autonomously find equilibriums (stable states) for optimal operation, when subjected to internal and external perturbations.

7.4.1 Model and Simple Example

By modelling the system's dynamic behaviour as a chemical reaction network, we can make use of analytical tools developed in chemistry to predict the behaviour of such systems, like for example Metabolic Control Analysis [Hof01] or Chemical Organization Theory [Dit07]. When modelling chemical communication in networks instead of encoding a deterministic state machine, or having a sequential program that processes an incoming packet, each network node contains a multiset $M(S)$ of a finite set of molecules $S = \{s_1, \ldots, s_n\}$ (each molecule represents a received packet added to the multiset; in this way the multiset models a chemical soup of molecules). Each node defines a set of reaction rules $R = \{r_1, \ldots, r_n\}$ expressing which reactant molecules can collide and which molecules are generated during this process. Such a reaction is typically represented as follows:

$$i1: C_i + X_i \rightarrow C_i + X_j$$

This reaction taking place at node i, consumes two molecules C and X from the local multiset, regenerates C and sends molecule X at node i to node j. In a simple two-node network topology, this example spans to the following reaction network (Figure 7-7):

$n1: C_1 + X_1 \rightarrow C_1 + X_2$ (according to rate r_1)
$n2: C_2 + X_2 \rightarrow C_2 + X_1$ (according to rate r_2)

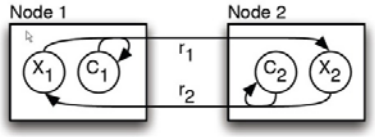


Figure 7-7

Whole protocol algorithms may be represented as sets of possible reactions in a reaction network. A received molecule (packet), which is placed into the multiset of the node, is likely to ignite, at some point in time, one or more (chemical) reactions. Delayed execution of reactions in the network (e.g. rule n1 after node 1 receives a new X molecule) as determined by an exact stochastic reaction algorithm, such as [Gib89], [Gil77], aim to enforce the "law of mass action" at the macroscopic level, which ensures that the reaction rate is proportional to the concentration (frequency) of reactant molecules in each multicet: $r_1 = c_1 x_1$ and $r_2 = c_2 x_2$. It can be shown that the overall reaction system spanned by the two local reaction rules strives towards the equilibrium (in a self-regulatory, i.e. adaptive way) where the number of X molecules in either node is inversely proportional to the number of the corresponding C molecules.

To illustrate with a simple example this model we present the simple implementation of a confirmed delivery protocol. Briefly a molecule refers to a piece of information, which maybe code, data, or code and data, and which appear in a network packet (molecule). The adopted symbolism is the following: $id[s_1 s_2 .. s_n]m$, where id refers to the execution point in the network, m reflects its concentration in the multiset, and s_i is a string representing code instructions or data. Instructions invoke transformations (packet re-writing), or reactions (when another suitable molecule is present in the multiset - determined by matching tags in position s_2). A list of operations in our prototype is given in Annex A. Figure 7-8 shows the sequence of chemical reactions and the reactant/resulting molecules in each node and how they perpetually lead to communication.

Chemical reactions model steps between protocol states. A set of possible reactions form a reaction network, representing a stochastic algorithm for which protocol states are represented by the concentrations of molecules; here the abundance of molecule X in the multiset). Thus, the protocol states in a finite state machine can vary in time.

When encoding protocol state machines in this way it is possible to relax the deterministic character of protocol execution due to the probabilistic character of reaction

execution, subject to external (environmental) pressures that influence the concentrations of molecules.

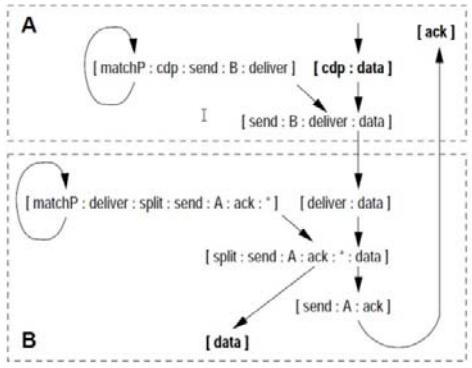


Figure 7-8

Adaptation emerges as the determinism dissolves, allowing the system to self-optimise and evolve towards one execution equilibrium, and then another as the environment changes. Environmental changes refer to resource availability and adverse conditions that might affect individual reactions (mutations, code attacks, etc) or molecular concentrations (communication conditions such as congestion, communication errors, etc), and which cannot affect inadvertedly anymore the system's stability (due to capacity the ability to evade determinism by following alternative paths in the reaction network).

On the other hand this encoding is more robust: when removing a molecule, the concentration only changes marginally and state information is not lost, only disturbed. Due to the law of mass action, the packet rate reflects the concentration of its originating chemicals. Thus, the packet rate itself, not the symbolic information inside the packet, is used to communicate state information among nodes and in doing so practically it re-programs itself. This rate coding scheme, akin to the nervous system [Day01], results in a higher resilience to the loss of packets. An extended example that highlights more adaptive aspects of chemical design is presented in Annex A.

7.4.2 Chemical Networking in INM

Instead of a specific solution to a network management problem, or a placeholder for management functionality, chemical networking suggests a paradigm for implementing management functions with self-regulating (and therefore adaptive) properties, inside the network, to sustain resource-functionality equilibrium. A typical example would be in-net distributed monitoring or network measurement protocols that self-adapt to resource availability and avoid congestion or other intrusive ("Heisenberg") effects [Roughan 05] on the network. The way chemical protocol design meets this goal (i.e. by suggesting the coexistence of network management operations and data services in the same traffic flows), adheres to the principle of the INM framework in 4WARD for designing management functions in an integrated way with service functionality. Continuous self-adaptation in the chemical networking paradigm is inherent in 3 specific objectives: (a) *self-healing*, whereby code is self-referential and able to regenerate its-self when needed, (b) *resource utilisation*, where system functionality is resilient to resource availability changes, and (c) *self-optimisation/protocol evolution*, whereby protocols can mutate and evolve online, by virtue of the inherent multi-stability that lets them glide from one equilibrium into another according to environmental conditions. In Annex A we compare the design of a protocol using CNPs with a counterpart from the literature.

The INM framework provides an ideal model for structuring "chemically" protocol functionality. In more practical terms, INM domains define the borders that confine chemical protocol communication between networked systems. Self-managing entities (SEs) are the placeholders of "molecule soups" where reactions take place. A set of interconnected reactions (a reaction network), implement an adaptive system/algorithm in an INM capability (MC); for example Quines that provide resilience of protocol code/state (regenerating accidentally or maliciously impaired code) and influx/outflux reactions that regulate resource occupancy (by amortizing molecule concentrations in the protocol state machines). The presence of certain molecule sets (e.g. Quines) in a soup, are monitored *objectives* through which the network operator is able to aggregate information about the state of services and the effects of their execution on the network. As, by nature, reaction networks are distributable across one or more soups (SEs) implementation can abide to all possible arrangements

highlighted in Section 7.2. Molecule soups are the high level (abstract) interface through which the network operator can influence (modify molecular concentrations, introduce/depreciate reaction rules, select dilution algorithms, etc) directly the dynamic behaviour or monitor the effects of protocol execution, which is more effective than indirect manipulation of static protocol parameters (today's practice).

7.5 Ensuring INM Stability with Built-in Verification of Configuration Changes

As the INM is running completely automatic and is solving any issues that might arise during the network's uptime, there is also a need for some sort of configuration mechanism. This mechanism configures the INM system both during the setup period but also during runtime. A fully automatic system might be unsuitable in certain situations, e.g. secure systems. However, manual interventions within the INM can be difficult to tackle as an automatic system can become inoperable by wrong configuration changes from an outside entity [Ada97].

For these reasons a configuration planning module is required. A configuration component facilitates manual intervention with new configuration. A prediction component prevents wrong changes, thereby ensuring network stability. The configuration planning module is considered to be running and available after the system has bootstrapped (during start-up the prediction module is unable to perform due to lack of real field data; only new configurations can be applied via the configuration component).

Configuration planning is done by an administrator (outside entity). The administrator interacts with the INM via an API which allows him to check the system state and set/modify and settings. Reconfiguration requests are processed by the configuration module, which invokes the prediction module for simulating the results of the proposed changes.

The prediction module implements a Markov chain of states, where a future state of the system depends only on the current state. The current network state comprises of all active nodes data containing:

- Number of live (active) connections
- Load of each node
- Number of loaded connections (active users that use most of the bandwidth)
- Average lifetime of a connection and delay of a package
- Alarms and errors (if there are any)
- Current policies (e.g. high-priority route policies such as VoIP which are to be kept alive under any circumstances)

The prediction module runs as follows.

Let there be $S = \{s_1, s_2, s_3 ... s_n\}$ where S is a set of given states in which the system can be in. The system can go from the state S_i to S_{i+1} with the transition probability p_{ij} (where i is the current state and j is the future state) that depends on the current state of the system. A stable network state is characterized by an equal or bigger number of live connections, at the same or a better connection speed while maintaining the current status quo between all network nodes. States are ranked in chronological order and computing the transition probability defines the probability of the system going from one stable state to another. The transition probability is the sum of all probabilities the system has any of its variables going in a different state. A so called transition matrix is constructed for each system variable.

For example the system constructs the node load matrix. The matrix for three nodes looks as follows:

$$P = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$$ Where lines and columns are set up like $P = \begin{matrix} A & B & C \\ X \\ Y \\ Z \end{matrix}$ and mean:

A – Node load will decrease by 10%

B – Node load will stay the same

C – Node load will increase by 10%

*A, B and C are allowed states the system can be in and X, Y and Z are the nodes for which the matrix is computed

This matrix shows that the system has a 33,3% chance of going from any state to the next one. The resulted probability of node X for example going from the current state B to state C is: $p_{BC} = \sum_{k=1}^{r} p_{Bk}\, p_{kC}$ Where r is the number of allowed states.

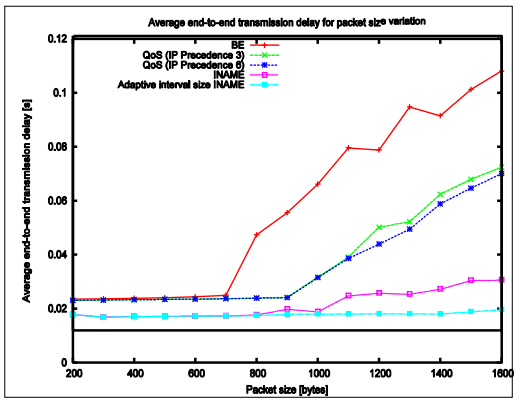
Current policies and rules apply as restrictions within each matrix.

If a smaller probability is detected the prediction module reports its findings and the network administrator gets notified.

This method allows for fast, on the fly and on demand simulations. It saves precious memory space as no network history is needed. Because of the fast computing time, this mechanism is suitable for merging networks. It is also beneficial for “mission critical applications” as it allows for quick access to resources in order to stay alive while having a minimal impact on the existing hardware infrastructure.

7.6 Self-adaptive QoS Management for VNets

In-Network Autonomic Management Environment (I-NAME) is an embedded self-adaptation algorithm, see [D4.2] that could be included inside the following MCs: INM ConfigurationCapabilities, INM RequestManagerCapability and INM CompositeMetric. It adapts user/manager/application requests to the network capabilities. VNet systems are major beneficiaries, because I-NAME is needed to negotiate the QoS parameters inside the established virtual networks. Moreover, if a virtual network aggregates similar applications in the same virtual space, it works for each particular application inside that space. Details are presented in Section 9.3.2



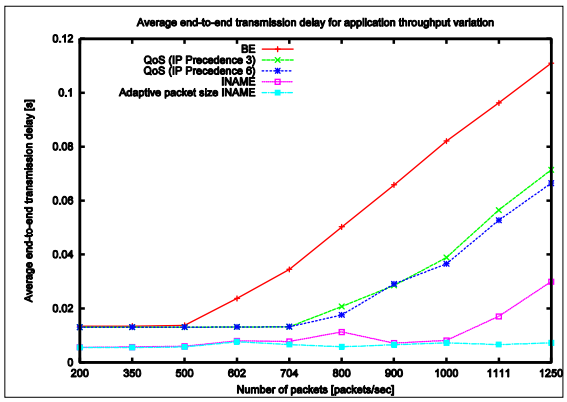


Figure 7-10: Average end-to-end transmission delay for the application packet size variation through adaptation process

Figure 7-11: Average end-to-end transmission delay for the number of packet transmitted variation through adaptation process

The benefits of I-NAME have been proved running simulations in QualNet. For example, considering the average end-to-end delay, the effects of packet size and interval between packets, Figure 7-10 and Figure 7-11 show that this self-adaptation algorithm is able to maintain the envisaged performances compared to best-effort, QoS IP precedence 3 and 6 approaches. The idea was to perform adaptation based on dynamic source fragmentation or coding.

7.7 Emergent-behaviour-based Congestion Control

Today's most well known and most used congestion control algorithm is realized in TCP – meaning in end-to-end manner at the transport layer. Within the scope of task TC45 we propose a new Congestion Control algorithm that works inside the network. Routers collaborate with each other in such a way, that they map their filling level to a certain frequency and report synchronize themselves along a path based on the highest frequency. This way the congestion level for a certain path is known to the other routers and based on this information a suitable path can be selected afterwards. A more detailed description of this approach is described in Section 9.4.3.

Self-adaptation Design Considerations

With the aim to reduce configuration and management overhead of the evaluated INM based congestion control approach, we applied a principle known as *emergent behaviour* to congestion control. In this context, a system is considered to have emergent behaviour in case each entity in the system applies simple rules (microscopic behaviour) which results in a more sophisticated behaviour of the overall system (macroscopic behaviour). This translates in case of a concrete implementation to the fact that the management control loops for each oscillator rely on input from the other oscillators (i.e. their neighbours). In addition the code to implement oscillators or the forwarding logic at the edge router can be separated from the corresponding router. Because of this two facts, we can identify option b) in Section 7.2 (Figure 7-1) as the most appropriate model to represent the approach described in this section.

7.8 Self-adaptive Routing in Wireless Multi-Hop Networks

Within the scope of TC45 we proposed a new self-adaptive routing protocol for Wireless Multi-Hop Networks called Adaptive Hybrid Routing Protocol (A-HRP). This approach combines the merits of the well known routing protocols and metrics, namely Optimized Link State Routing (OLSR) and Ad-hoc On-demand Distance Vector (AODV) and Expected Transmission Count (ETX) and Less Hop Count, into one common protocol. Moreover, it also introduces a new concept for adjusting the range/zone by each node to distribute topology information pro-actively via OLSR. The simulation results are very promising and show that a protocol that is able to adapt to different network situations is able to continuously perform above average in all situations and even outperform other approaches if the chosen network is moderate and not an extreme case. More detailed information can be found at Chapter 9.

Self-adaptation Design Considerations

Regarding a relation to a general INM self-adaptation scheme we can identify option a in Figure 7-1 as most appropriate. Reason for this is that each single entity acts on its own knowledge gathered by listening to the radio channel and exchanging local information (e.g. link qualities) with neighbouring nodes. Hence, the management control loops also needs input from other nodes. Due to this strong coupling it seems more natural to also fully embed the self-adaptation scheme within the MC.

7.9 Conclusions

Self-adaptation is a key feature for INM. This chapter presented multiple aspects of self-adaptation, initially discussing its characteristics in a generic manner, and then with focus on the specifics for each INM application. We presented design considerations for self-adaptation, and applied them to a few INM applications. Self-adaptation was elaborated in the context of self-organization of the management plane, avoidance of network implosion, search in self-organizing networks, INM applications as chemical processes, resource management, congestion control, and routing. Specific attention was given to self-adaptation of VNets (WP3) and GPs (WP5).

The chapter provides us with a wealth of knowledge regarding self-adaptation, its features, design aspects, and its applications, which is crucial for the implementation of INM capabilities in the future Internet.

8 INM Prototype Design

The main design elements of INM have been implemented into an integrated prototype, which provides a proof-of-concept of the INM architecture and algorithms. The prototype is based on a set of functions that have been selected following two criteria: (i) they show the benefits of the INM design; (ii) they can be composed to build a complete set of use cases for a demonstrator.

The prototype is written in Java and built on top of the OSGi framework [OSGi] for basic functions, such as remote method invocation, description and discovery of resources. Additionally, OSGi offers the possibility to dynamically instantiate new functions into the running network. This feature makes it possible to dynamically load management capabilities when new services are added into the network. In OSGi terms, a capability is then instantiated as a service, and different capabilities can be grouped together into a bundle.

The Organizational interface is defined as a generic interface, common to all capabilities: it mainly allows composition of objectives. The collaboration interface is instead a conceptual categorization of all the P2P interactions between capabilities. Communication between capabilities occurs on two levels: (i) through remote procedure calls (RPC) and (ii) through asynchronous communication (publish/subscribe of events).

Through the communication interfaces described above, capabilities are able to compose and maintain the network within pre-defined objectives, such expected quality of service or resource consumption. From an operator point of view, a console provides the graphical instrument to interactively operate the capabilities. The console in fact visualizes the actual values of key performance indicators and makes it possible to change the objectives. The console typically does not report a per-node view of the network, but it shows indicators and objectives that are aggregated from different sources. This level of presentation shows the feasibility of operating INM functions on a large scale also considering usability aspects of the interface towards the operator.

The naming of the capabilities in the prototype has been implemented with a simple scheme following the structure of the code. The ID of a capability is composed of the following tuple:

< capability-name; inm-runtime-id>

As *capability-name*, the name of the Java class is used; as *inm-runtime-id*, the address of the node is used. In most of the cases, a capability is implementing a management functions that is made available in the network. Therefore, in these cases it is sufficient to use only the name of the capability, omitting its location. The prototype contains therefore two functions implementing discovery of capabilities:

```
public Object[] getInmElements(String inm-runtime-id, String name-capability);
public Object[] getInmElements(String inm-runtime-id);
```

INM Platform

The INM platform is the set of libraries that provide basic interfaces to execute and integrate the INM algorithms. The interfaces are common to all the INM algorithms, implemented as capabilities, and offer the means to compose algorithms into complete management processes, like shown in Figure 8-1. The implementation of the INM components follows closely the design of the INM framework and is composed by two parts:

a) INM Runtime: defines run-time functions for INM and their implementation. No knowledge about the underlying OSGi specifics is required for an implementer of INM functions on top of the INM runtime.

b) INM Capabilities: defines the interfaces of individual management capabilities.

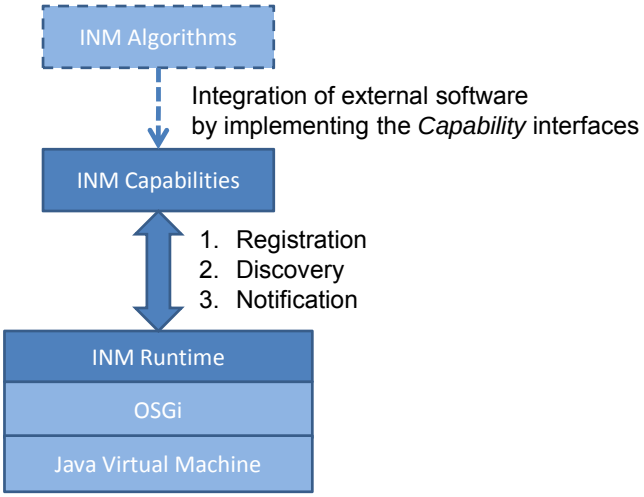


Figure 8-1: Mapping of algorithm design into INM Platform.

INM Runtime

The INM Runtime library provides the *Runtime* class. An instance of Runtime must be running on every node in the network. It provides the abstractions for INM of the underlying platform: these abstractions are based on INM specific methods and do not contain OSGi specific methods. The following functions are supported by the Runtime class:

1) Registration: every INM capability (implementation of an algorithm) must register to the INM Runtime. The registration makes it possible to discover that capability and to compose it with other capabilities, including the GUI.

2) Discovery: capabilities can be discovered through the Runtime class. Discovery is performed across all the different nodes.

3) Integration with capabilities. Alarms are handled by the Runtime class and delivered to the appropriate capability in the INM Domain. Triggers from monitoring to adaptation are also delivered through the Runtime class.

Communication between capabilities

The communication between capabilities occurs with two approaches:

- Method invocation between capabilities. The discovery method is used to retrieve a handle of the remote capability. The handle can be used for remote method invocation between capabilities. On the backend the handle is implemented though an OSGi proxy, which then implements the remote invocation.

- Publish/subscribe. This approach implements in general asynchronous communication and is used (i) to implement the “management by exception” paradigm of INM and (ii) to trigger INM capabilities, such as adaptation.

Example

The following use case gives an example of how INM capabilities are composed through the INM Runtime. The prototype can be decomposed in three functional parts: (i) graphical interface, which is the instrument given to operators for management, (ii) management algorithms for monitoring and adaptation, (ii) self-organization algorithms for re-configuration of the management algorithms. The use case is depicted in Figure 8-2.

Anomaly detection is a monitoring algorithm: it produces the correct series of test messages to indentify anomalies over the link (anomaly detection is discussed in Section 6.3.5). The algorithm works in a peer to peer fashion, communicating between two instances on different nodes: list of neighbours, and definition of test messages is provided by the INM Runtime. The analysis result of the anomaly detection is made available through the Organization interface; anomalies are published asynchronously as triggers for adaptation. In the example, congestion control is depicted as adaptation function to make changes in the network configuration.

Clustering introduces self-organization: it composes objectives and provides an abstraction layer for the graphical interface. Objectives of the anomaly detection, as well as the anomaly events, are grouped first inside a cluster. Consequently the congestion control capability and the graphical interface are not overloaded with information coming from the underlying INM capabilities and scalability is maintained even with large network settings.

An extensive description of the prototype interface and its document is planned in the final period of the project, and will be made available in a separated deliverable.

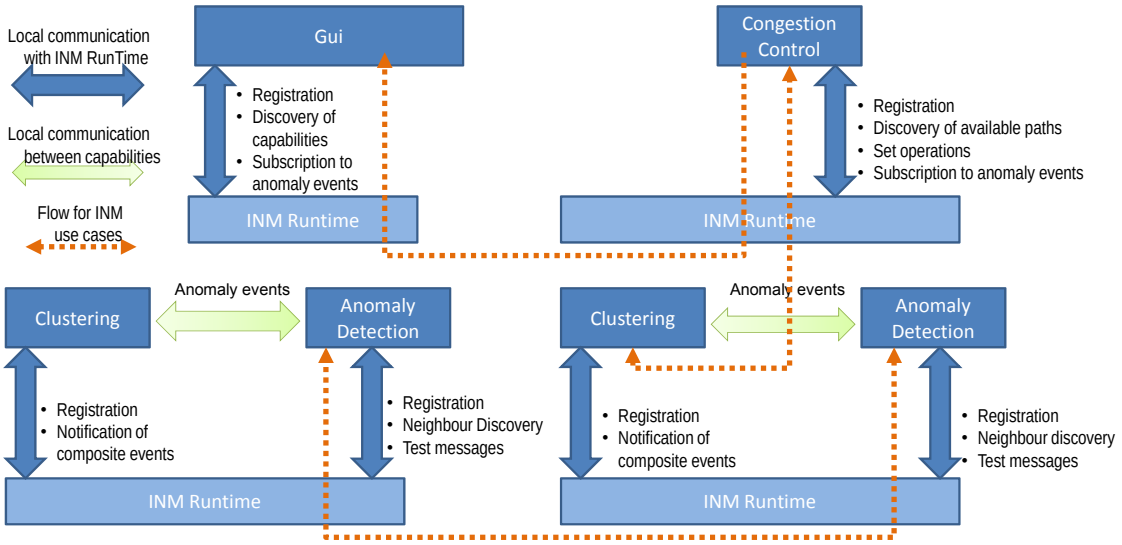


Figure 8-2: Composition of capabilities and their mapping to the prototype.

9 Integration between INM and other WPs in 4WARD

9.1 Integration of INM with 4WARD New Architecture Principles and Concepts (WP2)

WP2 is exploring the development of a design process for combining existing, or specifying and generating new networks with customized architectures. WP2 proposes a number of architectural constructs which aid them in their task, namely *stratum* and *netlet*.

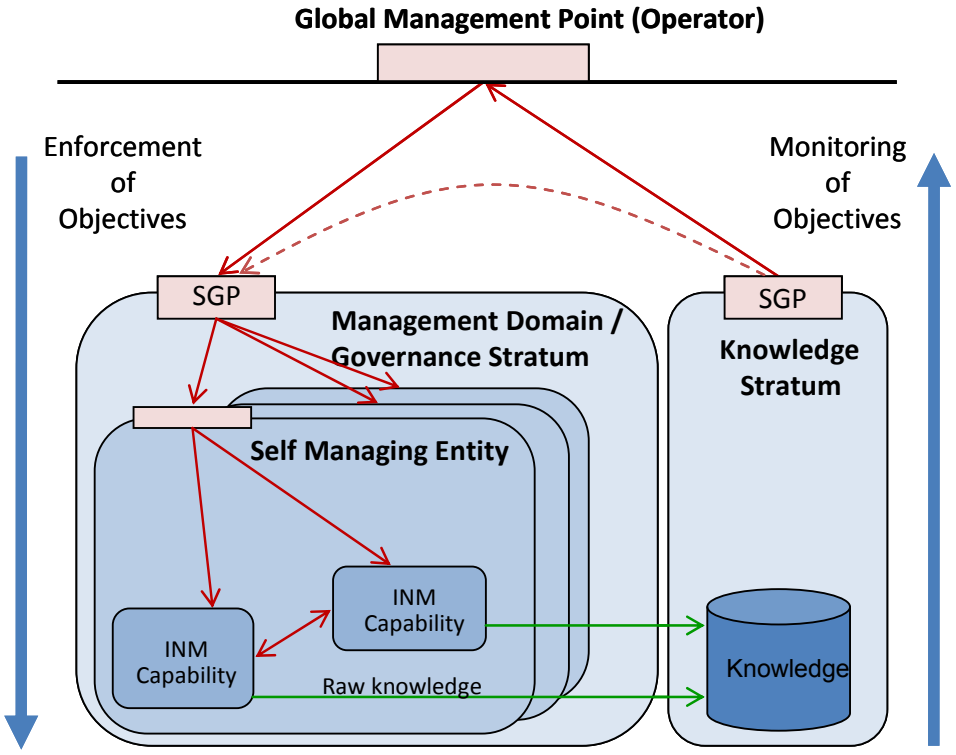


Figure 9-1: INM Relationship with WP2 Architecture

Figure 9-1 shows the mapping of INM to the Governance and Knowledge strata [D2.3]. The Governance stratum is related to a management domain and the set of Self Managing Entities which exists inside of this. Management objectives are pushed through the Stratum Gateway Point (SGP), these objectives are propagated downwards into the self managing entities and in turn down to the INM Capabilities which realise the objectives. The INM Capabilities which can realise one of the INM algorithms (monitoring or self adaptation) produce information or data, possibly in an unprocessed or unreasoned upon form. This information or data is then fed into the knowledge stratum which performs the processing or reasoning and produces knowledge. This knowledge can be propagated upwards to an operator or reused to manipulate the objectives via the governance stratum.

WP2 have also developed a design process which represents the workflow from a business idea as a starting point to the design of the suitable network and software models, which basically constitutes the blueprint of the network design. Figure 9-2 gives an outline of the design process and where the INM algorithms can be positioned.

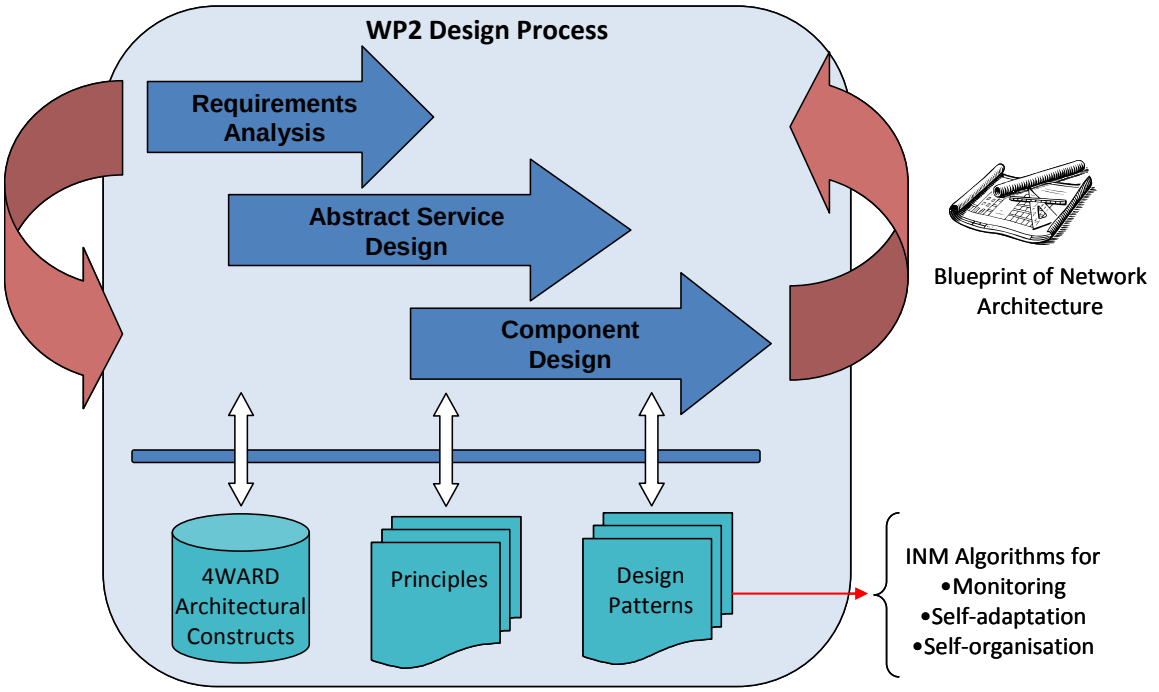


Figure 9-2: INM Relationship with WP2 Design Process

All stages of the design process use architectural constructs, principles and design patterns where appropriate when building their network blueprint. The management algorithms developed in WP4 are in essence design patterns for solving different monitoring and adaptation problems. These algorithms can be key components of the WP2 design patterns repository and can be integrated at the network design stage rather than a separate management concept at a later stage.

9.2 Considerations on the Use of NetInf Technology for INM (WP6)

Network management functionality produces a significant amount of information, through counters, events/alarms and configuration parameters that determine system state during operation. Regardless on the architecture of the system (centralised, decentralised, hybrid) this data needs to be accessed efficiently for analysis and correlation purposes. In this deliverable, we report on the use of NetInf infrastructure for distributed storage and sharing of management information, and on implementing scalable search on NetInf using an INM algorithm. The discussions on building a self-managing NetInf machinery are within the scope of the 4WARD D6.3 deliverable due in June 2010.

9.2.1 Storing Management Information as Information Objects

WP6 proposes a generic information-centric approach [Ohl09], where the identity of an information object (or any other characteristics thereof in the more general case) is used as a reference to retrieve it from any node that might store the full information object.

INM processes are implemented in a distributed manner, through co-locating service functionality and its embedded management capabilities. Such distributed management processes need efficient access and storage of information as a prerequisite for an efficient execution. Different types of management information required by processes within the INM framework were identified in [Bru08]. In order to understand which information-centric storage approach is suitable for storing particular types of management information, we conducted a detailed analytical study of existing information-centric storage mechanisms for wireless multi-hop networks, therein termed data-centric storage mechanisms.

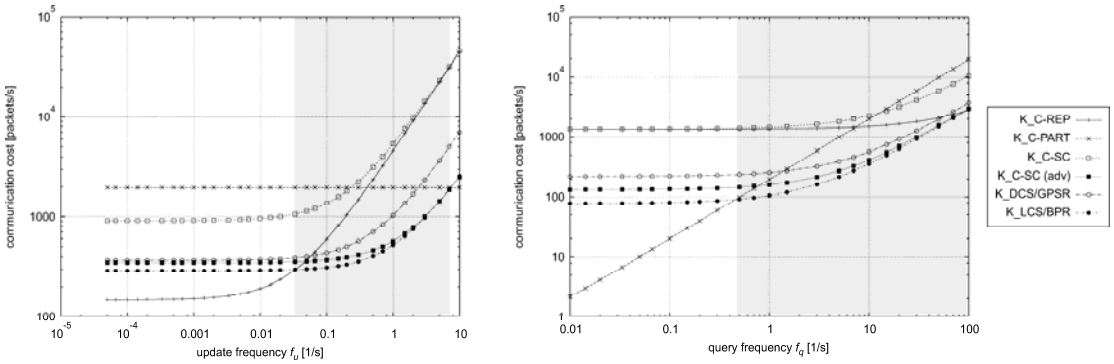


Figure 9-3: Communication cost of data-centric storage approaches as a function of update frequency (left) and query frequency (right).

Figure 9-3 shows two results that can be used to identify operational ranges of different data-centric storage mechanisms. The figure illustrates the characteristics of data-centric storage approaches in a multi-hop wireless network (in the presented case, a mobile ad-hoc network) as a function of update frequency (left side) and query frequency (right side), which are among the most important parameters that affect overall system performance. The communication cost indicate, for a typical scenario comprising 600 nodes on a 4 square kilometre area with each node having a communication range of 100 m and moving at speeds of up to 50 km/h (to model pedestrians and also cars), the number of packets required in a multi-hop wireless network for different values of update and query frequency.

The significant performance variations in the approaches presented are due to the different storage strategies employed. For instance, for highly dynamic management information items that are rarely queried on an atomic basis (such as the low-level bit error rate on a single wireless link), a cell-based data-centric storage approach without advertisement (abbreviated C_SC(adv) in Figure 9-3) might be the most appropriate choice. In contrast, for management information that is rather static (such as configuration parameters of a transport protocol for wireless multi-hop networks) and often queried by management capabilities in the vicinity of a specific network location, a cell-based replication approach (C-REP) might be the most adequate. These results can serve as input towards the use by INM of an adaptive NetInf Local Storage Engine and Cache Engine that adapts the selection of storage strategy to changing network conditions and information characteristics.

9.2.2 Scalable Metadata-Directed Search in NetInf

One important feature that NetInf must provide to users is scalable metadata-directed search for information objects. Using aggregation and threshold detection algorithms devised for INM, we can realise scalable one-time search and continuous search functionality. Algorithms can run on the Dictionary Node (DN) infrastructure of NetInf, which handles the distributed hash tables for name resolution. The main idea is to let the network itself do major parts of the work of finding and collating search matches in a distributed way using aggregation in a tree overlay of DNs.

When a NetInf user performs a search operation, an aggregation process is started within the Autonomous System (AS) by the nearest DN, which then initialises a global aggregation process of search results with participants from each AS in the network. Metadata-directed search will then take place concurrently on DNs all over the network, with results being passed from child to parent in the tree; see Figure 9-4 for an illustration.

If the user wants to do a one-time search, obtaining all objects in the network matching the query at a particular time instant, we use the echo algorithm [Lim01]. On the other hand, if the user wants to be continually updated on new matching information objects, we use GAP [Dam05]. In this way, the completion time for one-time search grows sublinearly with the system size, and for continuous search, there is a tradeoff between load (efficiency) and timeliness (accuracy) [Pal10].

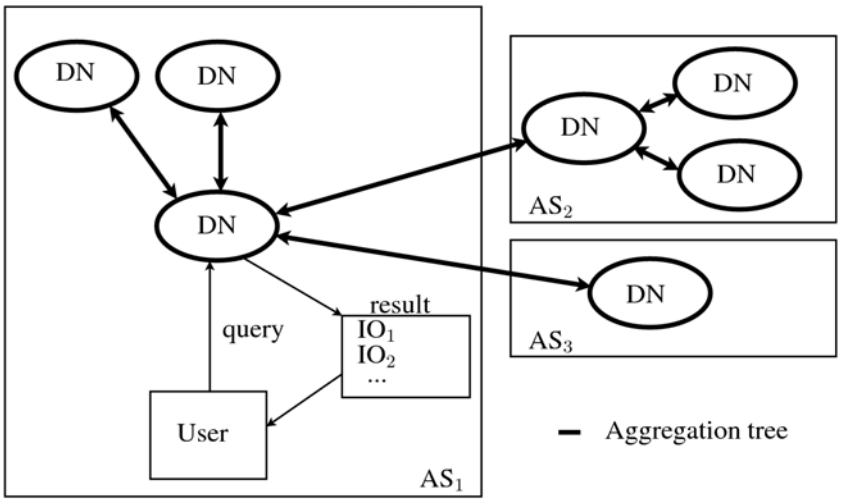


Figure 9-4: Example of metadata-directed search using tree-based aggregation.

9.3 INM Operations for Virtual Networks (WP3)

In the context of virtual networking, INM concepts apply in two places, as illustrated in Figure 9-5. First, INM provides management operation to the base infrastructure. Second, INM capabilities and functions are implemented in accordance with the network architecture running within a virtual network. Management capabilities in both places might need to interact, for the purpose of optimization. The INM boundaries and the behaviour at the boundaries concerning inter-operator issues can show its potential, shielding some information, and providing other information, for the sake of an optimal operation of the overall virtual networking environment.

There are several issues to be addressed. The decentralized operation of the infrastructure achieves a well balanced resource usage and concludes possible virtual node re-location. The virtual links are subject to resource management action in order to guarantee the bandwidth and QoS requirements. For both of these resource-related tasks, the current state and location of the resource are needed, which can be obtained by a situation framework in a fairly generic manner. Finally, the issues involved in fault management are of major importance in a carrier-grade virtual networking environment.

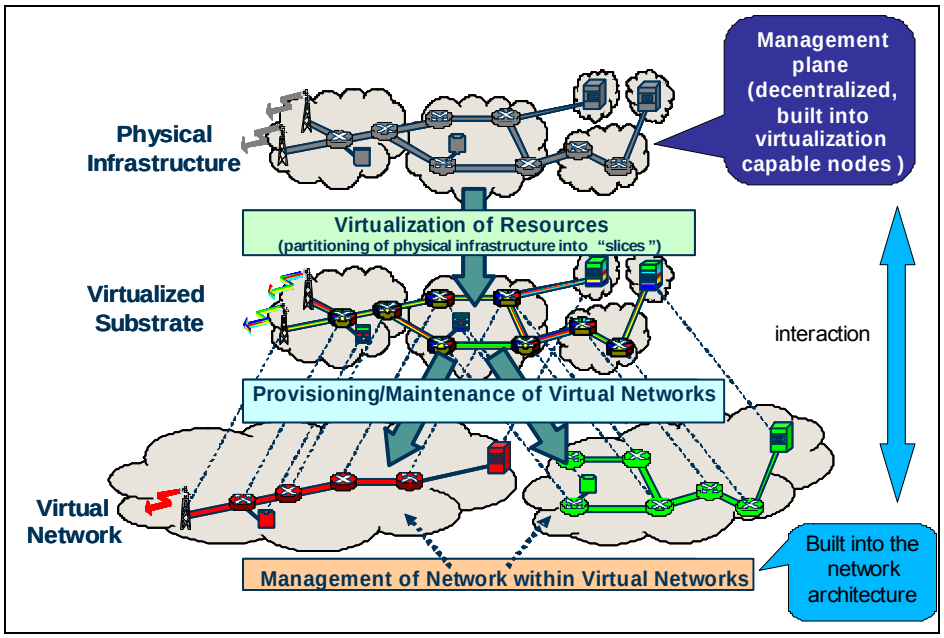


Figure 9-5: Overview of INM in Vnet

9.3.1 Decentralised Vnet Infrastructure Operations

The efficient use of physical resources is a key challenge in network virtualisation. To accomplish this task, the management of the physical resources should be transparent to the applications running within the virtual networks, and should be executed at runtime in order to deal with the variation on the load requests of different virtual networks. Traditional resource allocation schemes use offline, centralised and global view strategies to manage the use of physical resources. In contrast, INM proposes a runtime, distributed, local view approach to manage physical resources.

For the definition of the distributed employment, local view, and online features that are related to the reallocation of resources of virtual networks, some assumptions must be observed:

- The initial deployment of a virtual network and changes of the topology are addressed by the provisioning of the Vnet architecture described above.
- The reallocation of virtual entities must be as transparent as possible to the virtual network. In the current stage of this research, the reallocation of virtual entities is transparent with regards to information exchange, but not with regards to the reallocation process. There is no exchange of information between the virtual application running inside the moving entities, and the virtual managers of the physical entities that are involved in the reallocation process. However, the application must be interrupted during the reallocation process. Interactions at a later stage are foreseen, which complicates the process when negotiation needs to take place.

Reallocation of virtual entities can be triggered by special events, e.g. maintenance or exchange of physical entities, energy optimization, or traffic optimization. For traffic optimization the main objective of maintenance function is to identify the source virtual node that is generating a great amount of traffic to the destination virtual node. The optimisation – if possible - is done by moving the source virtual node from its physical device to another physical device near the destination virtual node.

Figure 9-6 illustrates this example. Three physical nodes are presented, which belong to a physical provider and labelled as PR-I, PR-II and PR-III. The virtual provider #2 has a physical connection across the three nodes, and in fact all the three physical nodes are executing the respective application (the light grey box inside each node). On the other side, the virtual provider #1 has undertaken a resource reallocation, and the respective application has been moved from PR-III to PR-II.

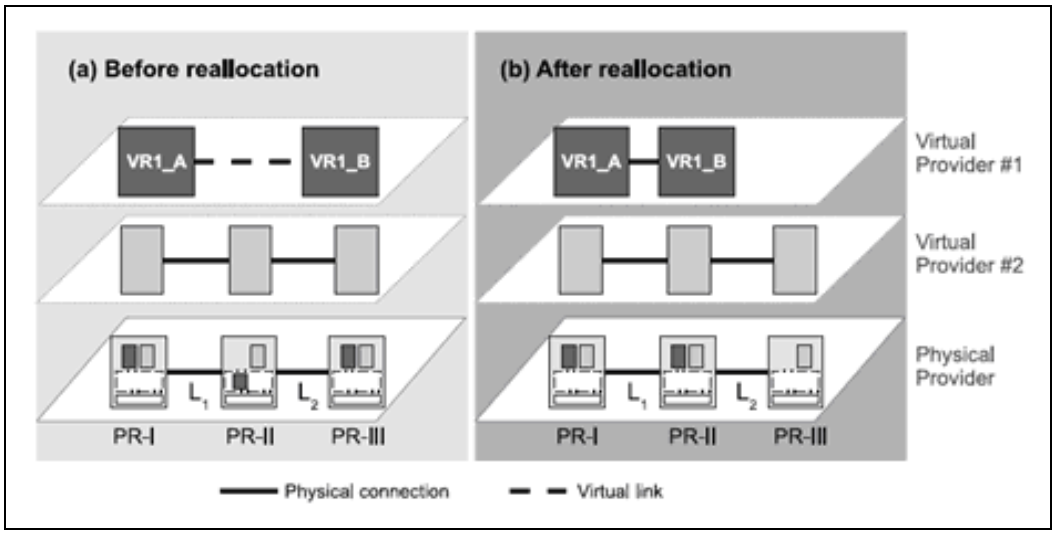


Figure 9-6: Relocation management with INM

9.3.2 Resource Management for VNET

Resource management is an INM operation that can enjoy an increase in the level of automation. With this respect, a scheme for automatic detection of dynamic network configurations and for the subsequent reaction is proposed. The scheme is supported through a set of specific INM capabilities that are collectively referred to as I-NAME (In-Network Autonomic Management Environment). To demonstrate I-NAME capabilities on providing end-to-end in-network resource management, we run time-critical test applications from a source node (SN) to a destination node (DN) under different network conditions: i) with I-NAME support, ii) over a best-effort network environment, and iii) based on QoS network layer classification. This demonstration is depicted in Figure 9-7, and the following abbreviations are used: SN=Source Node, DN=Destination Node, GP=Generic Path, GN=Generic Node, AP=Access Point, and BS=Base Station. Details are included in [D4.2] and in Section 7.6.

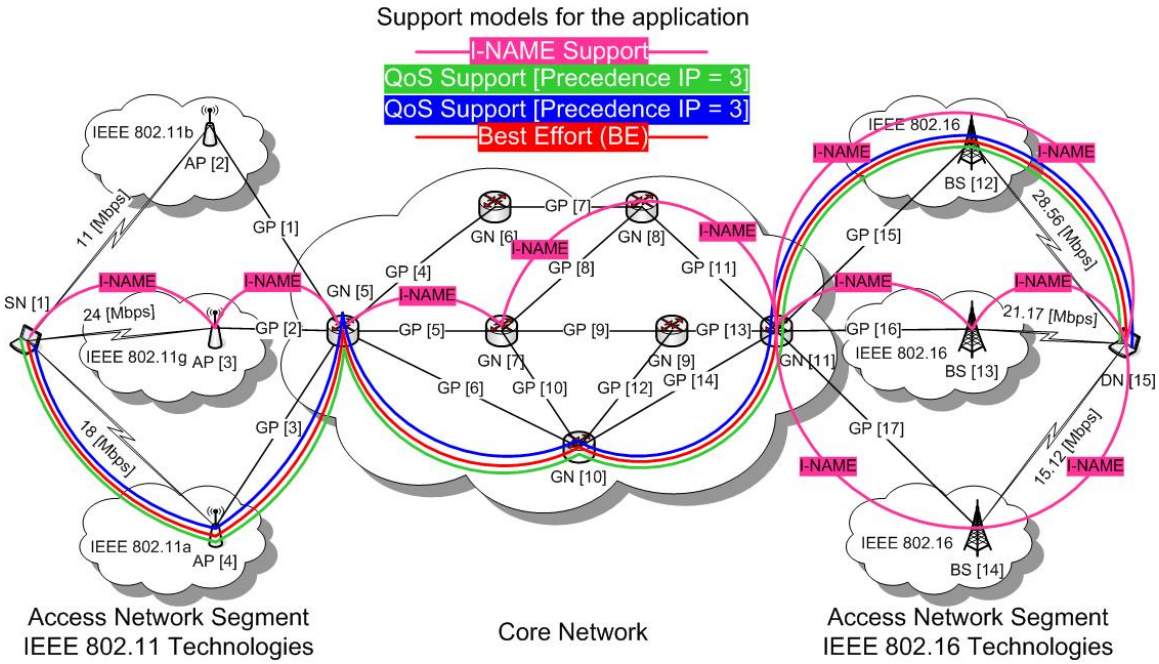


Figure 9-7: I-NAME performance evaluation scenario

We compare the results, in terms of average end-to-end transmission delay for the selected path in the network. In the first case (with I-NAME), resource management support is based on the QoS Profile message exchange between neighbour entities on the path form the SN to the DN. The QoS profile negotiates in each network entity a set of QoS parameters that synchronizes the application's request with the current capabilities on the path to carry the requested parameters. I-NAME model negotiates and indicates to the application the appropriate settings involved in the adaptation process. The performance evaluations proved the following conclusions:

1. Context-aware applications request specific resources in the network using QoS parameters and QoS weights to describe the application. Thus, optimal resources are allocated.
2. Network knowledge is useful to provide in-network management support for context-aware application. The obtained results are the network context and the best path.

I-NAME algorithm adds a resource management function to the INM capabilities, adapting user/ application requests to the network's capabilities. Vnet system generates virtual networks and INM negotiates the QoS parameters inside the established virtual network. Moreover, if a virtual network aggregates similar applications in the same virtual space, INM works for each particular application inside that space.

Figure 9-8 shows I-NAME role in INM and Vnets.

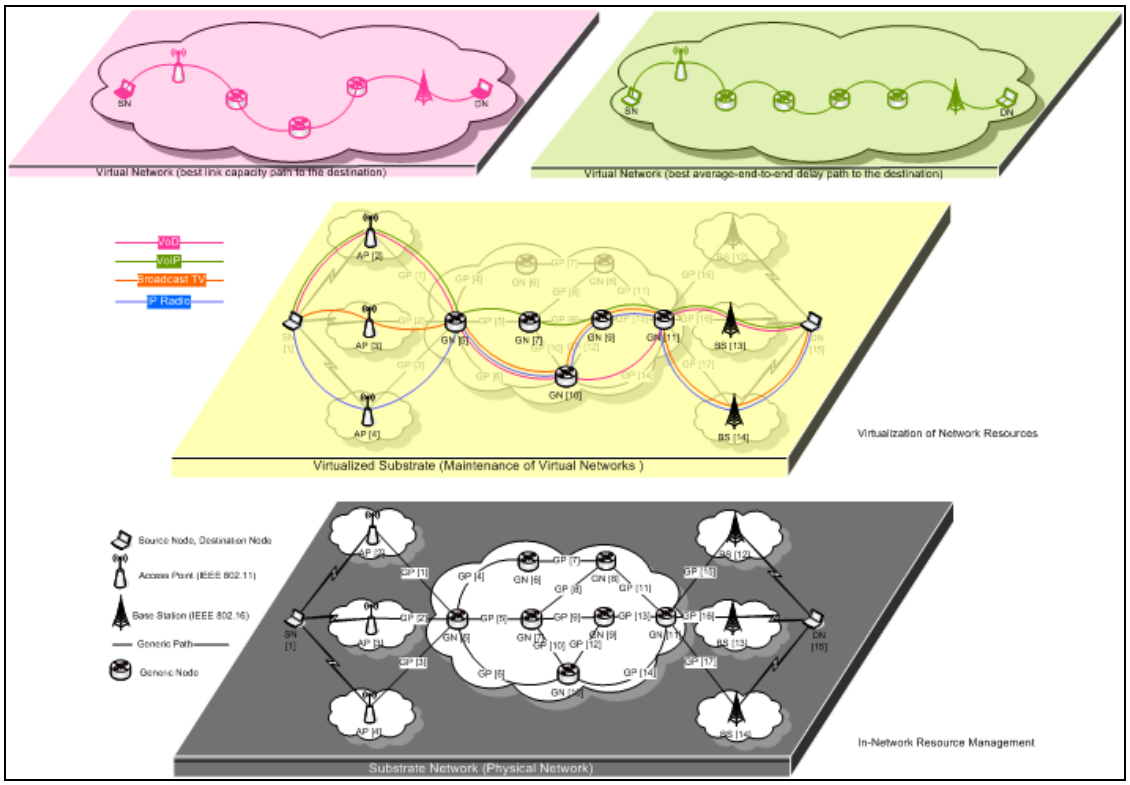


Figure 9-8: The I-NAME's role in INM and Vnet

9.3.3 Physical and Virtual Resource Awareness

As described previously, the efficient usage of physical resources is a critical requirement to enable network virtualisation. To be able to efficiently deal with physical resources, it is required that the virtualisation system is always aware about the state of each resource (used/available). From a higher and more general perspective, the virtualization system also needs to be aware of the virtualized resources, as those physical resources will be offered as virtualized resources to users.

The situation awareness framework introduced by WP4 is a concept that can be easily adapted for the purpose of network virtualisation and which enables such resource awareness. Via this framework, information about physical and virtual resources can be offered by network nodes that want to share these with other nodes. Such information sharing is possible with the deployment of directories, where nodes register themselves with the type of information they want to offer and with the location where this information can be received from (acting as provider). Upon a request of a node that wants to receive information about a certain resource (acting as consumer) the directories then provide a lookup service, responding with the location where this information can be retrieved from. Then, the provider and consumer nodes can directly communicate with each other and exchange the resource information either in a synchronous or asynchronous manner. In the synchronous case, a consumer will just receive a reply once, acknowledging its resource request. In the asynchronous case, the consumer subscribes to the information of the provider's resource and will continuously receive information about changes pertaining to this resource. The information exchange process is indicated in Figure 9-9.

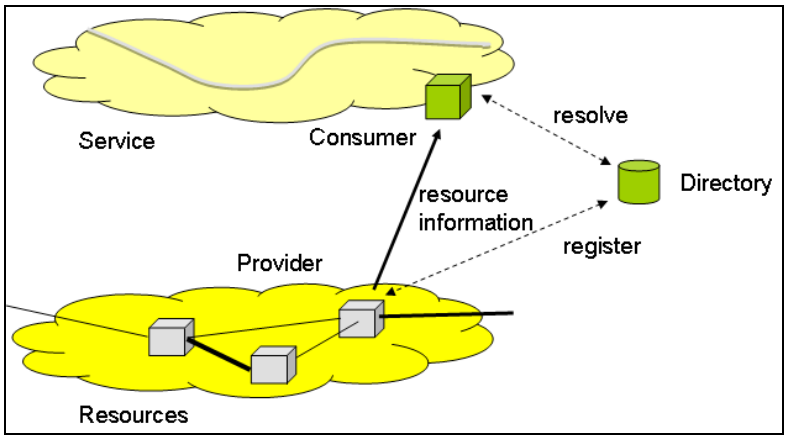


Figure 9-9: Resource Information exchange via the Situation Awareness Framework

The merit of using this approach for the purpose of network virtualisation is that nodes can request and receive information of different kinds of resources, e.g. physical or virtual, via standardised interfaces. Moreover, this information can be offered with different levels of details, e.g., aggregated resource information. This way, it is possible to obtain information that may be required to supervise offered virtualisation services.

9.3.4 INM Fault Management for Virtual Networking

Fault management in virtual networks is a relevant problem that occurs for infrastructure providers and other entities involved in network virtualization (Figure 9-10).

From the perspective of an infrastructure provider, detection of a fault is difficult, since it may not have any knowledge about the services that are running within the virtual network. Nevertheless, the fault detection mechanism can be based on some of WP4's work on anomaly detection. Once a fault is detected, an important question is whether or not to expose it. Many operators will naturally opt not to expose faults, while attempting to remedy such fault using self-healing mechanisms. Such mechanisms may make use of pre-provisioned backup resources or other resources that are provisioned on demand (similar to the basic provisioning step), or relocate virtual nodes to working infrastructure nodes. Note that in the case of using back-up resources, fault handling occurs at a much higher speed but requires a larger amount of resources beforehand. When there is no other way, a problem should be exposed to the virtual network provider, which knows its customers and should be able to determine which of the customers are affected.

Since the Vnet operator is mainly a broker for resources and an aggregator of business, it is mainly affected contractually when failures happen.

In the virtual operator space a failure can also be detected, as described before due to some occurring anomalies. In this case however, more knowledge is available about the network architecture and the services running within the architecture. When a problem is detected, at least the virtual resource that is affected by the problem can be identified. This situation can be signalled downward to the infrastructure provider to deal with it, or it can be handled on the service level through re-routing or other service-specific mechanisms.

Toward the end user we expect again to have some sort of service level agreement (SLA), where faults should be mentioned, and will most likely be handled contractually (as in today's networks).

Currently, a number of open issues remain:

- Is there a need for exchanging fault-related information between infrastructure provider and Vnet operator? Such exchange may help to fix fault situations faster, and without contractual involvement of the Vnet provider as a broker.
- What actions should be taken if the SLA to the end-user is affected?
- How does a service provider get involved in the fault management process of virtual networks? For example, with which aspects of fault management should an IPTV service provider be involved when using a Vnet operator for providing his service to end-users?

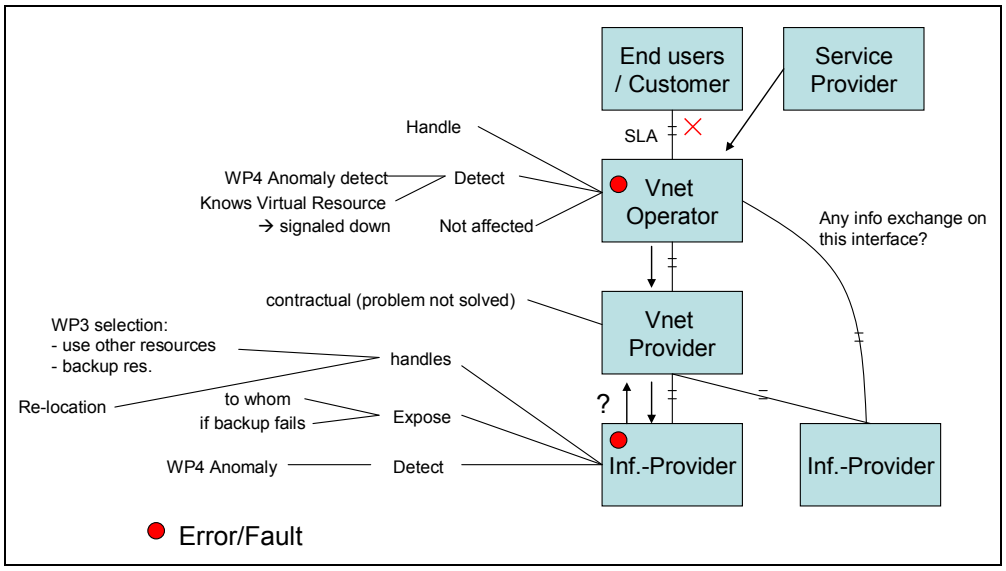


Figure 9-10: Fault Management in Vnet

9.4 INM support for Routing, Forwarding and Generic Paths (WP5)

In this section several applications of INM supporting routing, forwarding and Generic Paths are presented.

The basic application of INM for routing is the usage of **routing metrics**, which describe the routing paths quality. The usage of advanced routing metrics is especially important in case of adaptive routing based on network state, multi-path routing, wireless mesh networks and the cross-layer approach (Sections 9.4.1 and 9.4.4). The use of collaborative measurements, provided by INM differs significantly from simple path estimation approaches incorporated in present routing protocols.

Within the scope of INM, we use an **adaptive approach to distributed anomaly detection and fault-localization** (Section 6.3.5). In relation to GP, this approach can be used to detect network failures and at the same time provide measures for availability of network equipment. For improved network performance and service quality, measurements obtained from the proposed anomaly detection approach can be used as input to various GP operations, such as resource allocation, optimization and routing decisions.

The **Cooperation and Coding Framework** (CCFW, [D5.2]) has been developed to dynamically apply cooperation and network coding techniques in scenarios where they are beneficial. Monitoring of the environment checks whether all the requirements are fulfilled and whether the system can benefit from the resulting consequences. As a lot of local and remote

information is already available in the INM system, this information can be reused to feed the CCFW.

The basic principle of congestion control used in today's Internet relies on an implicit or explicit collaboration of end systems and routers. A new, **congestion control based on emergent behaviour** approach (Section 9.4.3) does not rely on the interaction between end-systems and routers and aims to reduce the required configuration and management overhead to a minimum. Another approach to congestion control for Generic Paths is based on network coding (Section 9.4.2).

A **distributed strategy for management of resources** of wireless mesh networks is concentrated on the efficient channel assignment and scheduling such that interference is minimized and the throughput maximized [D5.2]. It lies on a distributed approach for dynamic radio channel allocation of each node. Efficient mechanisms for the collection of network state information from the neighbouring nodes, within a so called network coherence time, are crucial for an efficient distributed management of the node's resources. To achieve this, INM may help thanks to its collaborative monitoring capabilities, thus enabling efficient *aggregation of network information* in space and time.

9.4.1 Path Quality Estimation in Wireless Networks

The real-time monitoring functions introduced by INM can play an important role in supporting the routing operations in clean-slate wireless networks. The novelty of the proposed approach is a separation of path monitoring from routing. There are several advantages of this approach:

- Real-time monitoring functions are used not only for routing but also for other purposes like anomaly detection, denial of service attacks, intrinsic network adaptation, etc. Due to the proposed approach the monitoring data are reusable,
- In contrast to relatively simple monitoring algorithms, implemented in routing protocols, INM monitoring operations are much more sophisticated and they can provide network state information in a more cost-effective manner and with greater accuracy,
- Separating monitoring from routing makes the routing protocols simpler, so they may focus on core routing operations such as maintenance of the routing tables,
- The monitoring functions might be tailored to different networking technologies, i.e., they can deal with monitoring of physical and link layers in a cross-layer approach making core routing operations independent of the specific networking technology used.

In routing protocols the selection of the "best" path (typically only one) for packets forwarding is done accordingly to some predefined parameter, named 'cost', or path metric. The path quality estimation procedure has to deal with the minimization of the measurement overhead, the reduction of the traffic volume related to the dissemination of the measurements and provide adequate accuracy and time resolution of the measurements. In present packet-switched networks the operations related to the path evaluation are relatively simple and the metric is typically a part of the routing protocol. Usually the path metric is a composition of the metrics of links which comprise the path. The per link path estimation approach has several benefits regarding implementation, overhead and the delay of path measurements. Depending on the estimated path parameter, a metric can be considered as additive (for example delay), concave (the minimum value, e.g. throughput) or multiplicative. The path metric can include the following parameters: number of hops (hop-count), packet loss, path delay, delay jitter, throughput, path load, path MTU (Maximum Transfer Unit) [Bau07]. At present the most popular metric is the hop-count metric. Unfortunately such simple metric ignores link load and a different bit-rate of links and because of that is unsuitable for adaptive and multipath routing. For advanced routing more sophisticated metrics have to be used to evaluate paths load. While in wired networks such estimation of path quality is relatively simple in wireless networks it is still an important problem. These networks require a specific approach due to a

complex behaviour of the physical layer and sharing of radio resources among many nodes. This problem has been extensively studied, and a plethora of possible solutions have been proposed. The most popular metrics for wireless mesh networks are: the hop count, Round Trip Time (RTT), Expected Transmission Count (ETX), Expected Transmission Time (ETT), Weighted Cumulative Expected Transmission Time (WCETT) and IEEE 802.11s Airtime [Bau07]. These metrics estimate paths quality, but without complete analysis of the complex physical layer phenomena. In such networks the link and path quality has to be estimated in a very complicated manner, taking into account not only the radio activity of the neighbouring (interfering) nodes but also the hidden and exposed terminal problem (so called self-interferences). In fact, the estimation of the path quality parameters should in take into account all underlying mechanisms, necessitating a cross-layer optimisation approach.

In the remainder of this section we will show an example of path quality estimation algorithm in wireless networks which has some novel properties in comparison with the existing algorithms. An algorithm, named Path Predicted Transmission Time (PPTT, [Yin06]) designed for wireless mesh networks, which is not dependent on any routing protocol is used in the presented case. PPTT enables the prediction of path delay on the basis of link measurements and intrinsic mechanisms modelling using the value of the traffic which will be injected into a path. This algorithm takes into account some cross-layer properties, such as radio channel occupancy level and the hidden terminal phenomenon. A new multipath routing protocol called LAMA (Load Adaptive MultipAth routing protocol), which is a combination of PPTT and the AOMDV protocol has been developed in the context of 4WARD. LAMA is using AOMDV to find multiple paths between the source and the destination. In the next step PPTT is used for the prediction of the end-to-end delay of these paths, and finally the best path in terms of delay, which is a function a path load and physical layer properties, is used for data forwarding. The benefit of this algorithm in comparison to routing approaches that are based on the hop count metric is shown in Figures 9-11 and 9-12. The figures show comparison of three routing protocols: AODV (single path with the hop-count metric) [RFC3561], AOMDV (multi-path with the hop-count metric) [Mar01] and LAMA. In all cases only one path has been used for traffic forwarding. For the simulations the CBR traffic has been used and the network topology was static. The evaluation of the path lengths shows that LAMA establishes longer paths than AODV and AOMDV (see Fig. 9-11), as expected. On the other hand, it is evident that AODV and AOMDV paths have similar path length, and their average value practically does not depend on the network load, as a result of the minimum hop count rule.

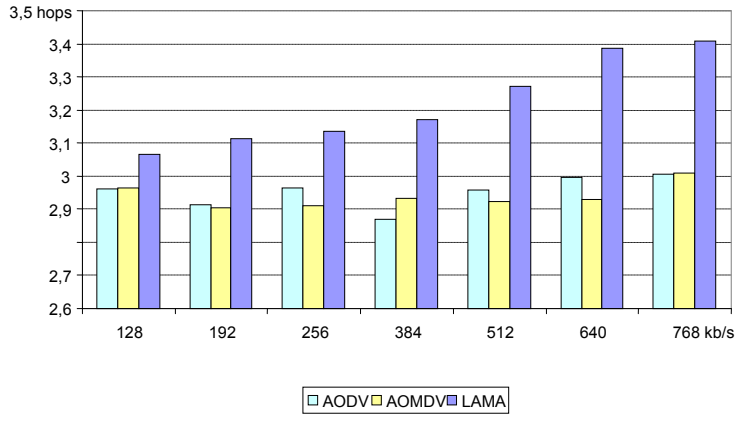


Figure 9-11: The mean end-to-end path length.

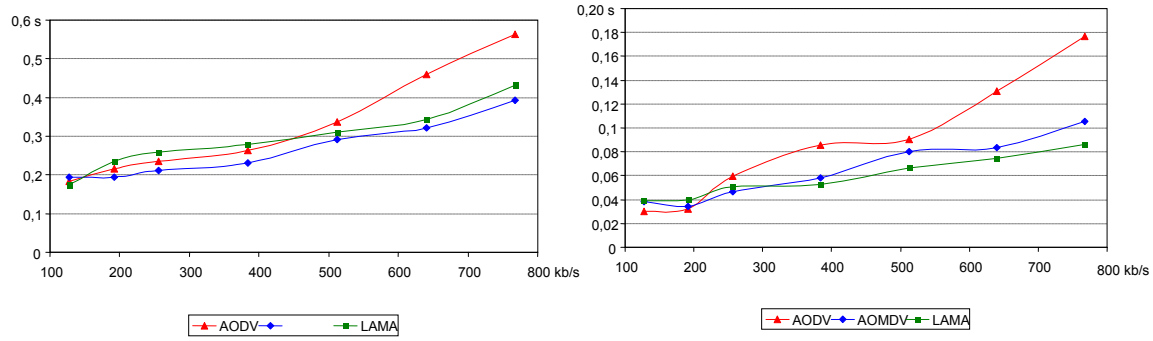


Figure 9-12 The end-to-end path delay (a) and delay jitter (b) as function of path load.

The comparison of AODV, AOMDV and LAMA end-to-end delay as a function of increased data traffic shows that AODV protocol demonstrates the worst behaviour, while AOMDV and LAMA delays are similar (see Fig. 9-12a), despite that LAMA paths are typically longer than AOMDV paths. The observation of the variability of delay jitter (Fig. 9-12b) confirms that the LAMA protocol uses paths with smaller fluctuations of the total transmission delay, i.e. the paths are less congested or have lower number of PHY retransmissions. The simulation presented above is an example of what can be obtained by the deployment of advanced, cross-layer-aware mechanisms for the prediction of the path quality.

9.4.2 Network Coding Usage for GP Congestion Control

A network coding (NC) scheme (described in [D5.2]) will be applied in order to obtain a better congestion control for GPs. However the employment of the NC techniques in practice requires a strong interaction with the network management and cross-layer techniques to acquire the link statistics of the network. Three principles are proposed for integration of the NC techniques into the GP architecture:

a) Instantiation of NC-capable GPs with multiple functioning modes: long term statistics acquired from INM is used to identify network topologies (link parameters) where NC solutions exist and congestion level is large enough. The nodes and links of the mentioned topologies are marked as appropriate for NC operations. GPs with NC capabilities are instantiated on the identified network topologies but NC operations are initiated only when it is necessary. Otherwise QoS-aware routing algorithms are employed in these GPs for the data transfer.

b) Dynamic NC application: link and flow characteristics are continuously monitored by INM are identified the situations when the short and medium term statistics of congestion level and links transfer characteristics make NC viable solutions.

c) Dynamic flow encoding: theoretical studies of NC operations consider that coding operations are performed continuously. Due to the burstiness of the real flows, data packets to be encoded are not present all the time in each node. On the other hand frequent activation/deactivation of the NC operations is not acceptable due to the signalling overhead. The proposed solution is the following: once the NC operations are activated coding operations are performed only in the moments when data packets are available for encoding, otherwise non-coded packets are transmitted, as the decoding algorithm is able to discriminate them.

The described integration of the NC mechanism into the GP architecture is proposed to be initially tested by the well-known butterfly topology, see Figure 9-13 and [Pol09].

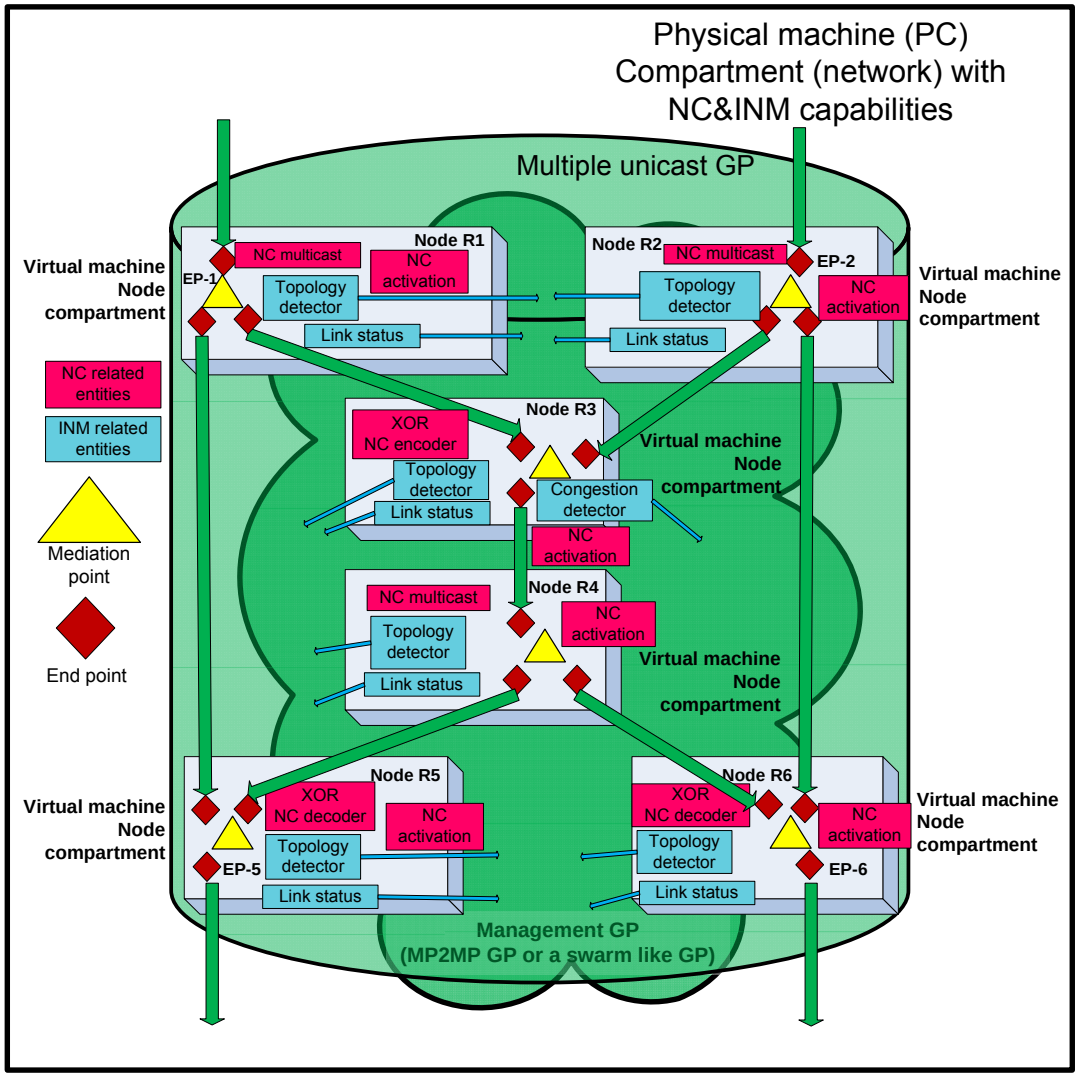


Figure 9-13: Integration of the NC mechanism into the GP architecture

The nodes implementing the coding network are represented by virtual machines instantiated in a physical machine which simulates a compartment with NC and INM capabilities. Sub-GPs implementing the butterfly coding network are represented by UDP connections. On the mentioned butterfly topology a multiple multicast GP is built, transferring unicast flows between the input endpoints, EP-1 and EP-2, and the output endpoints, EP-5 and EP-6. Activation of the NC operations is realized if the topology link parameters fulfil imposed conditions for transfer rates and delays and if congestions are detected for a longer time interval. Dynamic XOR based flow encoding is performed in node R3, and the decoding performed in nodes R5 and R6 is adapted to this type of NC operations. Exchange of the management information is performed on a specialized multi-point-to-multi-point GP (for example a swarm-like GP).

9.4.3 Emergent behaviour-based Congestion Control

The basic principle of congestion control used in today's Internet relies on the implicit or explicit collaboration of end systems and routers. It can be described as follows: In case the filling level of a routers queue reaches a pre-configured threshold, queue management strategies as e.g. RED are applied to drop packets in a random fashion. TCP connections of end-systems affected by these artificial packet losses reduce their sending rate to avoid the impending congestion. In case of UDP based communication it is up to the affected

application to implement a suited flow or congestion control principle. In fact the required interaction between end-systems and routers can lead to an unfair sharing of network resources. To address such issues, we focused on an INM congestion control approach which does not rely on the interaction between end-systems and routers. With the aim to reduce required configuration and management overhead to a minimum we applied a principle known as emergent behaviour to congestion control. More concrete, the emergent phase synchronisation phenomena of pulse-coupled oscillators, which is a well known phenomena in biology and physics. To apply this sync property to congestion control we started by:

1. *Associating each queue in a router with an oscillator* based on the Mirollo & Strogatz model [Mir90].

2. *Identifying the filling level of an queue with the corresponding oscillators frequency* (by using a linear function) [Kle09].

The applicability of the approach has been studied in the context of Multipath-Routing Scenarios in 4WARD's TC45. In Multipath-Routing, multiple alternative paths between a data source and sink are calculated which can be utilised for the actual data transfer. In our approach, each path calculated, defined a group of oscillators associated with the corresponding router queues (e.g. g1, g2 and g3 in Figure 9-14] (a)). The oscillators in each group are assumed to be coupled during or after path calculation.

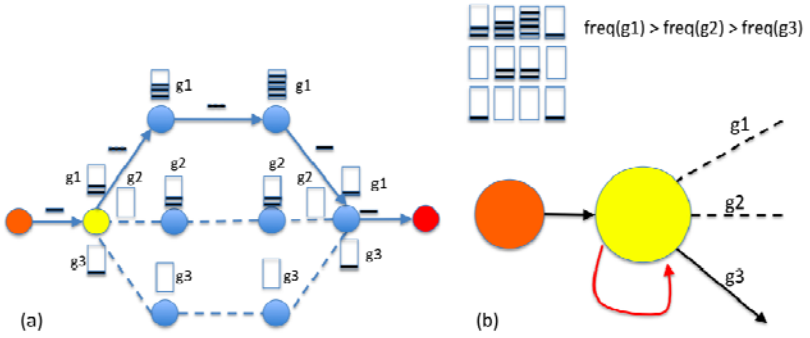


Figure 9-14: Least Congested Path First example

The idea behind the approach is that based on the synchronisation property of the corresponding oscillators, the highest oscillator frequency in g1, g2 and g3 defines the frequency of the whole group. Consequently, in case of congestion as shown in Figure 9-14 (b), the edge router (coloured yellow) is able to determine the path with the lowest maximal filling level for the next packet by comparing the frequencies of its oscillators.

As the result of our evaluation work (Annex C) we can state:

1) *After a appropriate selection of oscillator related parameters the sync property emerges in the target multipath scenarios. We evaluated scenarios with a path length up to 100 routers.*

2) For the addressed scenarios, the sync property also emerges in case of frequency variation. The resulting group frequency is the frequency of the fastest oscillator.

9.4.4 Self-adaptive Routing in Wireless Multi-Hop Networks

Within the last decade there has been a growing interest to enable communication, meaning Internet-Access, via wireless multi-hop networks for daily life. Such types of networks are usually referred as wireless mesh networks (WMNs), which consists out of mesh clients and mesh routers [Aky05]. Typically mesh routers have minimal mobility and form the backbone of the WMN whereas mesh clients often are laptops or cell phones and thus also exhibit mobile behaviour.

Evolutionary WMNs were derived from the research area of mobile ad-hoc networks (MANETs) and thus could be seen as one possible research branch of MANETs. However, from today's point of view one could also argue vice-versa. A MANET could be a special case of a WMN where the whole network just consists out of mesh clients. Anyway, regardless of these viewpoints there clearly is a relation in between those two types of networks, which also shows the fact that communication protocols designed for MANETs are used for WMNs as well.

In the meantime various routing protocols and metrics have been proposed for WMNs which try to optimize the usage of network resources assuming static network conditions [Cou03] [RFC3626] [Awe04]. Unfortunately, these approaches just address one type of network entity, namely the mesh router. Even 802.11s just assumes four key scenarios focussing on static network conditions with further restrictions such as a maximum network size of 50 nodes and routing on layer 2 called path selection [Con05].

As WMNs will be also part of a Future Internet as one possible access technology it is obvious that there is an urgent demand to deal with those problems. The network condition might rapidly change, e.g. due to data downloads of multiple simultaneous connections (data traffic) or due to changing mobility patterns of users. This implies that it is not suitable to just pre-configure each network device with a routing solution that seems appropriate. There is a need to adapt the routing behaviour of each mesh node based on the given network situation.

Within the scope of our work in 4WARD's TC45 we investigated the possibility to adapt the behaviour of each single mesh node. The goal of this adaptation is to optimize the usage of available network resources and not to optimize the throughput of a single data flow. To achieve this goal we expect all nodes to collaborate with each other based on the guideline that the interest of the group is more important than the interest of a single entity. However, the most difficult problem here is the nature of wireless communication. Contrary to wired networks, communication costs in wireless networks are incomparably more expensive. It is inappropriate and unaffordable to waste available bandwidth with further control overhead as the limited bandwidth will be needed for data transmission. Hence, the only information that can be used to adapt the behaviour of each mesh node is any kind of information that can be gathered by listening to the radio channel - meaning information of the one hop neighbourhood.

In network simulator 2 (ns-2) [NS2] we simulated different metric and protocol combinations in various network conditions showing that there indeed is a huge performance gap in between protocols and metrics depending on the given scenario. Results show that on the one hand the smaller the network size, the lower the network density and the lower the mobility the better performs the Optimized Link State Routing protocol OLSR [RFC3626] using the Expected Transmission Count (ETX) [Cou03] as metric whereas with increasing network size, increasing node density and higher mobility the better performs Ad-hoc On-demand Distance Vector protocol AODV [RFC3561] using Less Hop Count as metric. Based on these results we present a new approach called Adaptive Hybrid Routing Protocol (A-HRP) that enables routing based on different protocols and metrics, namely OLSR, AODV, Minimum Hop Count and ETX. Moreover, A-HRP allows each node to adaptively change the range/zone of distributing its OLSR routing information (Annex D). Figure 9-15 shows an example how a packet could be routed via A-HRP.

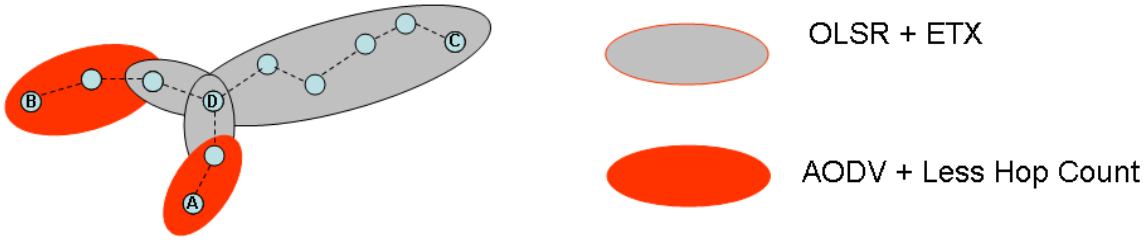


Figure 9-15: Exemplarily routing via A-HRP

Our simulations show that A-HRP can achieve an overall good performance for all scenarios and outperforms all other approaches in scenarios that do not reflect extreme cases such as a completely static network, a high number of nodes, or high node density. By adaptively changing the broadcasting zone we directly influence the knowledge of the network topology which in return leads to an adaptation of the used routing protocol and metric (Annex D).

10 INM Results vs. Related Projects

Other research projects overlap in scope and approach to some extent with our work. In deliverable [D4.2], we compared our work to several of those projects (including US projects). This section complements [D4.2] focusing on the four most relevant projects and discussing them in more detail.

10.1 SOCRATES

SOCRATES - Self-Optimisation and self-ConfiguRATion in wirelEss networkS

www.fp7-socrates.eu

The SOCRATES project is supported by the European Union under the 7th Framework Program, and will run from January 1, 2008 until December 31, 2010. The project investigates the application of self-organisation methods, comprising mechanisms for self-optimisation, self-configuration and self-healing, as a promising opportunity to automate the planning and operation of wireless access networks in general, and the 3GPP Long Term Evolution (LTE) radio interface in particular.

The SOCRATES project has investigated a list of use cases and has derived requirements of the different components of self-organisation from them. Assessment criteria for methods and algorithms for self-organisation are also developed. The SOCRATES framework has been formulated to establish a basic set of ideas, rules, principles and boundary conditions for the development of concrete self-organisation. Furthermore, the framework has identified interrelationships and dependencies between the different use cases, to ensure that potential conflicts are regarded during solution development from the beginning. Currently, the project is developing use-case oriented methods and algorithms for self-optimisation, self-configuration and self-healing. Simulations are being conducted to assess their performance. As far as the feasible OAM solution is concerned, the SOCRATES project keeps the option open, whether a centralized approach, a distributed approach or a hybrid approach should be followed.

In short, there are clear relevance and similarities between the 4WARD and SOCRATES projects. Both 4WARD (in the context of INM) and SOCRATES projects deal with the topics of efficient network management and self-organizing networks (SON). One principal objective of the two projects is to effectuate substantial operational expenditure (OPEX)

reductions by diminishing human involvement in network operational and management tasks, while optimising network efficiency and service quality. There are, however, also clear differences between the two projects. While 4WARD has taken a top-down approach by heading on architectural design and principles for management entities, SOCRATES has taken a bottom-up approach by scrutinizing the potentials for concrete use cases first. While 4WARD has a longer time horizon and aims more at framework and clean-slate vision, SOCRATES has a shorter time horizon and is close to standardization (3GPP LTE) and products. While 4WARD's scenarios are quite general and the developed schemes are universal, SOCRATES' use cases are concrete and the developed algorithms are expected to achieve specific radio-related effects. Thus, the contents of the two projects are in some senses complementary.

To achieve synergies of the results from the two projects, it should be investigated whether the SOCRATES use cases have posed some concrete requirements which need fine tuning in the INM concepts and schemes, whether INM framework can be used for long-term development of SOCRATES management architectural issues, and, whether some INM distributed communication mechanisms/algorithms can be used as SOCRATES use case solutions.

10.2 AutoI

AutoI - Autonomic Internet

http://ist-autoi.eu

AutoI is an FP7 project funded in Call 1. I started in January 2008 and is due to finish June 2010. The objective of AutoI is to create a communication resource overlay with autonomic characteristics for the purposes of fast and guaranteed service delivery. Given this objective, AutoI will design and develop, based on a well-defined methodologies, an open software infrastructure and tools that enable the composition of better (fast and guaranteed) services in an efficient manner and the execution of these services in an adaptive (Autonomic form) way. To this end, it introduces Virtualisation of network resources and Policy-Based Management technique to describe and control the internal logic of the services, utilising Ontology-based information and data models to facilitate the Internet service deployment in terms of programmable networks facilities supporting the next generation of internet.

AutoI has defined two scenarios to drive the efforts of the project. The first scenario focuses on limited features of the virtualisation plane, management plane, service enablers plane and knowledge plane. The scenario traces the dynamic starting of a virtual network, deployment of services based on user context and the modification of this network based on user context. The second scenario increases the autonomic capabilities of the network, by introducing self-configuration and self-optimisation. One of the primary outputs of the project will be a set of open source tools that can be integrated into existing networks to increase the level of autonomic behaviour.

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

The service orchestration offered by AutoI may be of interest to 4WARD as similar techniques could be used when combining the 4WARD Self Managing Entities. Also the information model developed in AutoI as a means of abstracting itself from the underlying network may also be beneficial when the 4WARD INM Capabilities collaborate with each other with respect to heterogeneous underlying resources.

From a perspective of 4WARD offering outputs which may benefit AutoI; the monitoring and self adaptation algorithms would add value. They have the potential to be

deployed both at the network level and also at a virtual level (where AutoI performs majority of its work). The INM concept of co-design (designing the management and service in tandem) and the INM capability architectural component has the ability to augment the AutoI services, increasing their level of adaptability.

10.3 ANA

ANA - Autonomic Network Architecture

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

ANA is an FP6 funded project in Situated and Autonomic Communications (FET initiave). It aims at exploring novel ways of organizing and using networks beyond legacy Internet technology. The ultimate goal is to design and develop a novel network architecture that enables flexible, dynamic and fully autonomic formation of network nodes as well as whole networks.

The scientific objective in ANA has been to identify fundamental autonomic network principles, and build, demonstrate, and test such an autonomic network architecture, based on a model that scales in time and in a functional way; that is, the network can extend both horizontally (more functionality) as well as vertically (different ways of integrating abundant functionality) and change over time.

The challenge addressed has been to come up with a network architecture and to fill it with the functionality needed to demonstrate the feasibility of autonomic networking, which extends the passive neutrality of the Internet with a dynamic one that permits for an evolution even of the network's core elements and standards.

In ANA new requirements like support for data flow management (e.g., monitoring), security (in a broad sense), and network virtualization mechanisms (e.g., p2p, indirection) are easily and ubiquitously integratable in the architecture adhering to the premise that a functionally scaling network is the basis for an evolving network to include the various self-* attributes such as self-management, self-optimization, self-monitoring, self-repair, and self-protection. To meet this goal the notion of a *compartment* is paramount in the ANA architecture as a means of contextualizing and organizing functionality and services. By analogy the semantics of compartmentisation appear in 4WARD as well (GP compartment in WP5 and INM domain in WP4), however in more refined contexts. In ANA there is no distinction between the type of services offered in different compartments, and so one is free to define a contextual structure on a per service basis as opposed to per *type* of service. The INM framework in 4WARD goes one step further in specifying how services should be structured in order to retain generalness respect interfaces and interact with each other inside the architecture so as to avoid and ad-hoc solutions. This ensures that there is a way of maintaining local as well as global knowledge in a system, which is fundamental for monitoring and tracing feature interactions among competing services or functions in a system.

Through an Internet de-construction approach ANA explored the trends of functional atomization, diffusion and sedimentation that can lead to a replacement of the currently static layering of network functionality. Atomization refers to the decomposition of current layered networking software in smaller units that can be recombined. Functionality that belonged to one layer beforehand will be used in the future at arbitrary places (i.e. diffusion) inside a protocol stack or heap, requiring the introduction of autonomic organisation principles. Finally, sedimentation denotes the tendency of services to form compounds either with other required services or with the service clients. Again in 4WARD and specifically in the INM framework for management functions this process is taken one step further by identifying and classifying the different roles of atomic functional units and specifying placeholders in the framework's skeleton where their deployment is meaningful. In addition in INM attention has been given to classify and target the algorithmic research work to specific management objectives and operations (more focused), while in ANA the algorithmic work is more "open-ended" aiming

mainly to exemplify the use of the benefits and power of autonomic concepts developed of the project.

All in all, one may argue that ANA and 4WARD have similar objectives, and analogous research strategies. However, while ANA focuses more on a foundational work and in this process takes a de-compositional and investigational approach, 4WARD on the other hand takes a more in-depth engineering approach to provide structure for building concrete functionality. In this respect it may be perceived that the two projects have necessarily complementary objectives.

10.4 EFIPSANS

EFIPSANS - Exposing the Features in IP version Six protocols that can be exploited/extended for the purposes of designing/building Autonomic Networks and Services

http://www.efipsans.org/

EFIPSANS is a 7th Framework project addressing Objective 1.1 The network of the future. The project has started in Jan 08 and is running for 3 years. It investigates self-management features for the Future Internet with particular focus on IPv6 networks, where it looks at appropriate IPv6 enhancements and architectural extensions that enable the implementation of autonomic behaviours.

The project has been defining a Generic Autonomic Network Architecture (GANA) as unified reference model for autonomic/self-managing networks applicable to both evolutionary and clean-slate approaches. GANA provides a basis for standardizable specifications for autonomic behaviours. Core GANA concepts themselves are protocol agnostic, but they can be instantiated with specific autonomic control mechanisms and protocol functions, where the project looks in particular at functions of IPv6 and suitable extensions thereof.

GANA distinguishes four hierarchical levels of abstraction, namely protocol, abstracted network function, node/device and network level, for management and manageability aspects. Autonomic Manager Components are designed regarding hierarchical, peering, and sibling relations among each other. They co-operate with each other for the establishment of self-managing features of the network and exercise the autonomic control of associated Managed-Entities (MEs).

GANA is meant to address the problems of complexity in network management. GANA incorporates design principles for network-intrinsic management and defines constraints and boundaries for network-intrinsic management. GANA supports in implementing conflict-free decision-making-processes that control autonomicity/self-management behaviours within a node/device and at the network level and provides interfaces to users/operators of autonomic networks for setting Network-level Objectives.

Work on GANA in EFIPSANS has similar objectives as 4WARD INM, but they are executed on a different level. EFIPSANS has been contributing on a hierarchical architecture for network management that helps to model context and decisions of management operations. 4WARD complements to this research in two directions. From one side the framework has been focusing on design aspect of management capabilities and their integration with the service. On the other side, 4WARD has also provided a rich set of specific management functions in a distributed architecture as well as their evaluation. Both projects can therefore benefit from cooperation and pooling of their efforts for the creation of momentum.

11 Discussion

The main achievements and results of WP4 (during the second year of the project) towards the management of the future Internet can be summarized as follows.

INM Framework

- We have finalized the INM framework, which is an enabler of INM functions. Three main elements, namely, management capabilities, self-managing entities, and management domains were defined, from which complex management functions can be constructed.

INM Algorithms

- We have developed a set of INM algorithms for situation awareness that address real-time monitoring of network-wide metrics, group size estimation, topology discovery, data search and anomaly detection.
- We have also investigated algorithms aspects in the context of distributed network monitoring. Specifically, we have evaluated different algorithmic approaches to distributed aggregation and we have developed and evaluated stochastic models of tree-based aggregation under churn.
- We have developed secure versions of our algorithms for monitoring of network-wide metrics.
- We have developed a set of INM algorithms for self-adaptation, giving special attention to self-adaptation mechanisms required in the context of other areas of research within 4WARD, such as VNets (WP3) and GPs (WP5).
- We have developed self-adaptive solutions for the self-organization of the management plane and for aiding configuration planning.
- We have developed a framework for designing self-adapting network protocols inspired in models for chemical processes.

INM Prototype

- Currently, we are implementing an INM prototype. It will serve as a proof-of-concept for the INM approach, the framework and selected algorithms. The prototype will also serve as a basis for a demonstrator.

Dissemination

Based on results from work in this WP since the start of the project, 33 papers have been published in international conferences, journals and magazines, such as IEEE Transactions on Network and Service Management, IFIP/IEEE International Symposium on Integrated Network Management, and IEEE ICC. In addition, 11 are currently under review.

Furthermore, INM work is highly visible in the network management research community. Results are disseminated in the key conferences and events presenting technical papers and keynotes.

An outstanding example of the prominent participation of WP4 in top conferences in the area of network management is the last IFIP/IEEE Symposium on Integrated Network Management (IM 09) (http://www.ieee-im.org/2009/) IM is the main venue for original research in the area of network management. Its last edition was held in NY, USA in June 2009. It included technical sessions, application sessions, a mini-conference, panels, tutorials and several co-located workshops. The contributions by 4WARD partners to IM this year include four papers in the technical sessions (out of a total of 41), a paper in the mini-conference and

a poster. In addition, Dr. Marcus Brunner gave a keynote speech on In-network Management in the workshop on Management of the Future Internet (ManFI).

WP4 representatives have also met other EU-funded projects to discuss their work in network management. For instance, WP4 participated in the Joint EMANICS, AutoI, Self-Net Workshop on Autonomic Management organized by Professor George Pavlou (UCL) in April 2009. The goal of the workshop was to bring together researchers working on autonomic management in European projects. The workshop included 14 talks (including one by 4WARD) and had some 40 participants, most of them from academia.

Finally, WP4 has also disseminated some key results in the Future Internet Summer School, where Professors Danny Raz and Rolf Stadler gave a course on INM in July 2009.

12 Terminology

Anomaly detection	Analysis of measurements deviating from normally observed behaviour.
Atomic INM algorithm	An INM algorithm that cannot be decomposed into smaller parts without losing functionality.
CAPEX	Capital Expenditures.
CLQ	Cross-Layer Quality of Service is a management capability with two approaches. The bottom-up performs periodical measurements on top of the MAC sublayer, whilst top-down realizes resource allocation (as an INM function). In case of emergencies, CLQ could replace the existing schemes and support hop-by-hop transport between nodes.
Collaborative fault-localisation	Isolation of abnormal behaviour to certain network components.
Co-Design	Style of designing management functions in conjunction with service functions.
FCAPS	Fault and Configuration Management, Accounting Management & User Administration, Performance and Security Management.
Generic Aggregation Protocol (GAP)	Generic Aggregation Protocol. Distributed algorithm that provides continuous monitoring of global metrics.
Global Management Point (GMP)	High-level entry point via which a network is managed in terms of high-level objectives and according to the INM paradigm.
INM	In-Network Management.
Management Capability (MC)	The building blocks for composing any basic and any more complex management functions from management algorithms.

Management Domain	Specific view on a set of self-managing entities, either structural or functional, providing access to a restricted set of management functions only.
NATO!	"Not All aT Once!", a statistical probability scheme and algorithms for precisely estimating the size of a group of nodes affected by the same event, without explicit notification from each node, thereby avoiding feedback implosion.
OPEX	Operational Expenditures.
Remote Procedure Call (RPC)	A style of inter-process communication that can be used to implement distributed INM functions, e.g. on the level of management capabilities.
Representational State Transfer (REST)	A style of software architecture based on resources and a simplified set of functions to access such resources in a distributed way.
Self-adaptation control loop	An algorithm or a portion of an algorithm within an MC that implement self-adaptation functionality for an INM algorithm.
Self-Managing Entity (SE)	A component of a system that is self-managed by objective and can autonomously perform a series of management-related tasks, e.g. self-configuration and self-healing.

13 References

[Ada97] A. Adas, “Traffic Models in Broadband Networks”, IEEE Communications Magazine, July 1997.

[Aky05] Ian. F. Akyildiz and Xudong Wang, "A Survey on Wireless Mesh Networks", IEEE Communications Magazine, vol. 43, no. 9, s23-s30, Sept. 2005.

[Alo07] Rafael Del Hoyo Alonso et al., "Neural Networks for QoS Network Management", Computational and Ambient Intelligence, Springer Berlin / Heidelberg, 2007, pp. 887-894.

[And09] F. Andersen et al., “Impacts of Integrated Network Management on OPEX and CAPEX in Future Internet Scenarios”, 4WARD Technical Report, July 2009.

[Awe04] B. Awerbuch, D. Holmer, and H. Rubens, “High throughput route selection in multirate ad hoc wireless networks”, in Wireless On-Demand Network Systems (WONS), January 2004.

[BBF69] Broadband Forum TR-069 Amendment 2, CPE WAN Management Protocol v1.1, Dec. 2007.

[Bau07] R. Baumann et al., “A Survey on Routing Metrics”, ETH, TIK Report 262, 2007.

[Bra02] D. Braginsky and D. Estrin, “Rumor routing algorithm for sensor networks”, In Proceedings of the First Workshop on Sensor Networks and Applications (WSNA), Atlanta, GA, October 2002.

[Bru08] M. Brunner and F.-U. Andersen, “Opportunities, Requirements and Challenges for Storing Network Management Information in a Decentralized Way”, IEEE Workshop DANMS 2008 at IEEE Globecom 2008, New Orleans USA, 2008.

[Cho99] H. K. Choi and J. O. Limb, “A behavioral model of web traffic”, in Proceedings of the 7th Annual International Conference on Network Protocols, Washington, DC, USA, 1999, pp. 327.

[Clem06] Alex Clemm, “Network Management Fundamentals,” Cisco Press, 2006.

[Coh09] Reuven Cohen and Alexander Landau, "Not All At Once!, A generic Scheme for Estimating the Number of Affected Nodes While Avoiding Network Implosion”, Infocom'2009 mini-conference, Rio di Janeiro, Brazil, Apr, 2009.

[Con05] W. S. Conner, “IEEE 802.11 TGs Usage Models”, IEEE P802.11 Wireless LANs, Document IEEE 802.11-04/0662r16, Jan. 2005.

[Cou03] D. De Couto et al., “A High Throughput Path Metric for Multi-Hop Wireless Routing,” ACM Mobicom Conference, September 2003.

[Dam05] M. Dam, R. Stadler, “A Generic Protocol for Network State Aggregation”, Proc. Radiovetenskap och Kommunikation, Linköping, Sweden, 2005.

[Day01] P. Dayan and L.F. Abbott, “Theoretical Neuroscience”, MIT Press, 2001.

[Dit07] P. Dittrich and P. Speroni di Fenizio, “Chemical Organization Theory”, Bull. Math. Bio. 69 (2007), no. 4, 1199–1231.

[Dud09] D. Dudkowski, “Co-design patterns for embedded network management: Supporting the creation of large-scale distributed management systems”, submitted for publication, 2009.

[Dud09b] D. Dudkowski, “An Analytical Study of the Communication Cost of Data-Centric Storage in Mobile Ad-Hoc Networks”, submitted for publication, 2009.

[Dud09c] D. Dudkowski et al. , “Architectural Principles and Elements of In-Network Management “, Mini-conference at the 11th IFIP/IEEE International Symposium on Integrated Network Management (IM 2009), Long Island, New York, USA, June 1-5, 2009.

[D2.2] “4WARD Deliverable D-2.2: Draft Architectural Framework”, María Ángeles Callejo (editor), FP7-ICT-2007-1-216041-4WARD / D-2.2.

[D2.3] “4WARD Deliverable D-2.3: Architectural Framework”, María Ángeles Callejo (editor), FP7-ICT-2007-1-216041-4WARD / D-2.3.

[D3.1.1] “4WARD Deliverable D-3.1.1: Virtualisation Approach: Concept”, Stephan Baucke (editor), FP7-ICT-2007-1-216041-4WARD / D-3.1.1.

[D4.2] “4WARD Deliverable D-4.2: In-network management concept”, Giorgio Nunzi (editor), FP7-ICT-2007-1-216041-4WARD / D-4.2.

[D5.2] “4WARD Deliverable D-5.2: Description of Generic Path Mechanism”, Thorsten Biermann (editor), FP7-ICT-2007-1-216041-4WARD / D-5.2.

[D6.2] “4WARD Deliverable D-6.2: Second NetInf architecture description”, Matteo D’Ambrosio (editor), FP7-ICT-2007-1-216041-4WARD / D-6.2.

[Fie00] R.T. Fielding, “Architectural Styles and the Design of Network-Based Software Architectures”, Ph.D. dissertation, Dept. of Computer Science, Univ. of California, Irvine, 2000.

[Fär02] J. Färber, “Network game traffic modelling,” in Proceedings of the 1st workshop on Network and system support for games, New York, NY, USA, 2002, pp. 53–57.

[Gib89] M.A. Gibson and J. Bruck, “Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels”, J. Phys. Chem. A 104 (2000), no. 9, 1876–1889.

[Gil77] D.T. Gillespie, “Exact Stochastic Simulation of Coupled Chemical Reactions”, J. Phys. Chem. 81 (1977), no. 25, 2340–2361.

[Gua09] Lucas Guardalben et al., “HISKY: A Cooperative Hide and Seek Discovery over In Network Management”, Technical Report, Institute of Telecomunications/Portugal Telecom Inovação, December 2009, available at http://www.av.it.pt/ssargento/pub_report.html

[Hass05] G. Hasslinger, S. Schnitter and M. Franzke, The Efficiency of Traffic Engineering with Regard to Failure Resilience, Telecommunication Systems Vol. 29/2, Springer (2005) 109-130.

[Hass08] G. Hasslinger and S. Kempken, “Applying Random walks in structured and self-organizing networks” , PIK'2008, Special Issue on Self-organizing networks, Saur Verlag, Vol.. 31 (2008) 17-23.

[Hass09] G. Hasslinger and T. Kunz, ”Challenges for Routing and Search in Dynamic and Self-organizing Networks”, Proc. 8th AdHoc-Now, Sept. 2009, Murcia, Spain, Springer LNCS 5793 (2009) 42-54.

[Hei99] W. Heinzelman, J. Kulik, and H. Balakrishnan, “Adaptive protocols for information dissemination in wireless sensor networks”, In MobiCom’99, Seattle, WA, August 1999.

[Hof01] J.H.S. Hofmeyr, “Metabolic Control Analysis in a Nutshell”, Proc. 2nd Int. Conf. Sys. Bio., 2001, pp. 291–300.

[Hua06] P. Huang and T. Hwang, “On new moment estimation of parameters of the generalized gamma distribution using its characterization,” TJM, vol. 10, no. 4, pp. 1083–1093, 2004.

[Int00] C. Intanagonwiwat, R. Govindan, and D. Estrin, “Directed diffusion: A scalable and robust communication paradigm for sensor networks”, In MOBICOM, pages 56-67. ACM, 2000.

[ITU00] ITU-T: Recommendation M.3010 Principles for a telecommunications management network. (2000).

[JSN09] Java-Source.net, “Open Source Workflow Engines in Java,” http://java-source.net/open-source/workflow-engines, 2009.

[Kep03] J.O. Kephart and D.M. Chess, “The vision of autonomic computing”, IEEE Computer Magazine, January 2003, pp 41-50.

[Kle09] M. Kleis, “Sync: Towards Congestion Control based on Emergent Behavior”, Presentation at 9th Joint ITG and Euro-NF Workshop "Visions of Future Generation Networks" (Euroview2009). Available: (http://www3.informatik.uni-wuerzburg.de/euroview/2009/data/slides/Session8-Kleis-slides-handout.pdf) .

[Kre09] G. Kreitz, M. Dam, and D. Wikström, “Private Information Aggregation in Large Incomplete Networks,” KTH Technical Report, 2009, Available: http://www.csc.kth.se/~mfd/Papers/mpc_incomplete.pdf.

[Kum06] P. Kumar, “Probability distributions conditioned by the available in formation: Gamma distribution and moments,” Comput. Math. Appl., vol. 52, no. 3-4, pp. 289–304, 2006.

[Kwi08] R. Kwitt and A. Uhl, “Image similarity measurement by Kullback-Leibler divergences between complex wavelet subband statistics for texture retrieval,” in Proceedings of the 15th IEEE International Conference on Image Processing, San Diego, CA, 2008, pp. 933–936.

[Lim01] K.-S. Lim, R. Stadler, “A Navigation Pattern for Scalable Internet Management”, IEEE/IFIP International Symposium on Integrated Network Management, 2001.

[Mar01] M. K. Marina, S. Das, “On-demand Multipath Distance Vector Routing for Ad Hoc Networks”, 9th International Conference on Network Protocols, 2001, pages 14-23.

[Mey08] T. Meyer et al., “Robustness to Code and Data Deletion in Autocatalytic Quines”, Transactions on Computational Systems Biology X. Volume 10/1974, December 2008.

[MINTS] The Minnesota Internet traffic studies www.dtc.umn.edu/mints/references.html.

[Min09] Chiara Mingardi et al., “Event Handling in Clean-Slate Future Internet Management”, 11th IFIP/IEEE International Symposium on Integrated Network Management (IM 2009), Long Island, New York, USA, June 1-5, 2009.

[Mir90] R.E. Mirollo and S. H. Strogatz, “Synchronization of pulse-coupled biological oscillators”, SIAM Journal on Applied Mathematics vol. 50, no. 6, pp. 1645-1662, Nov. 1990.

[Mof02] Object Management Group: Meta Object Facility (MOF) Specification, Technical report, April 2002. http://www.omg.org/docs/formal/02-04-03.pdf.

[New03] M. E. J. Newman, ”The structure and function of complex networks,” SIAM Review, vol 45, no. 2, pp. 167–256, 2003.

[NS2] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/.

[Ohl09] B. Ohlman et al., First NetInf architecture description, FP7-ICT-2007-1-216041-4WARD / D-6.1, http://www.4ward-project.eu/index.php?s=file_download&id=39.

[OMG09] Object Management Group, “Business Process Modeling Notation (BPMN) Version 1.2,” http://www.omg.org/spec/BPMN/1.2, Jan. 2009.

[OSGi] “About the OSGi Service Platform – Technical Whitepaper”, June 2007, Available: http://www.osgi.org/wiki/uploads/Links/OSGiTechnicalWhitePaper.pdf.

[Pal10] K. Palmskog et al., “Scalable Metadata-Directed Search in a Network of Information”, submitted to Future Network & Mobile Summit, Florence, Italy, 2010.

[Pol09] Z. Polgar et al., “Preliminary Implementation of Point-to-Multi-Point Multicast Transmission Based on Cross-Layer QoS and Network Coding,” in 17th International Conference on Software, Telecommunications & Computer Networks IEEE SOFTCOM 2009, Split-Hvar-Korkula, 2009, pp.131 – 135.

[Rom07] R. Romeikat et al., “A Policy-Based System for Network-Wide Configuration Management”, Proceedings of the 18th World Wireless Research Forum, Helsinki, Finland, June 2007.

[RFC3561] Ad hoc On-Demand Distance Vector (AODV) Routing: RFC 3561.

[RFC3626] Optimized Link State Routing Protocol (OLSR): RFC 3626.

[Rou05] M. Roughan, “Fundamental bounds on the accuracy of network performance measurements”, ACM SIGMETRICS, Banff, Alberta, Canada, June 06 - 10, 2005.

[Sha02] M. Shaw,”Self-healing: Softening Precision to Avoid Brittleness”, Proc. 1st Workshop on Self-healing Systems, 2002, pp. 111–114.

[Sri01] C. Srisathapornphat et al., “Sensor information networking architecture and applications”, IEEE Personal Communications, 8:52-59, 2001.

[Ste09a] R. Steinert and D. Gillblad, “An initial approach to distributed adaptive fault-handling in networked systems,” SICS, Kista, Sweden, Rep. T2009:07, 2009.

[Ste09b] R. Steinert and D. Gillblad, “Distributed detection of latency shifts in networks,” SICS, Kista, Sweden, Rep. T2009:12, 2009.

[Ter07] W. Terpstra et al., “BubbleStorm: Resilient, probabilistic and exhaustive P2P search”, Proc. ACM SIGCOMM, Kyoto, Japan, 2007.

[Var09] C. Varas and T. Hirsch, "Self Protection through Collaboration Using D-CAF: A Distributed Context-Aware Firewall," Third International Conference on Emerging Security Information, Systems and Technologies, 2009.

[Vin08a] S. Vinoski, “Serendipitous Reuse,” IEEE Internet Computing, Jan./Feb. 2008, pp. 84–87.

[Vin08b] S. Vinoski, “Demystifying RESTful Data Coupling,” IEEE Internet Computing, Mar./Apr. 2008, pp. 87–90.

[Vin08c] S. Vinoski, “Convenience Over Correctness,” IEEE Internet Computing, July/August 2008, pp. 89–92.

[Wuh08] F. Wuhib et al., “Decentralized Detection of Global Threshold Crossings Using Aggregation Trees.” Computer Networks 52 (2008) 1745–1761.

[Wuh09a] F. Wuhib, R. Stadler, M. Dam: “Gossiping for Threshold Detection”, the 11th IFIP/IEEE International Symposium on Integrated Network Management (IM 2009), Long Island, New York, USA, June 1-5, 2009.

[Wuh09b] F. Wuhib et al., "Robust Monitoring of Network-wide Aggregates through Gossiping", IEEE Transactions on Network and Service Management (TNSM), Vol. 6, No. 2, June 2009.

[Yin06] S. Yin et al., “Traffic-aware Routing for Real Time Communications in Wireless Multi-hop Networks”, IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing, 5-7 June 2006.

[3GPP 32.816] 3GPP TR 32.816 V8.0.0, Study on management of Evolved Universal Terrestrial Radio Access Network (E-UTRAN) and Evolved Packet Core (EPC), Dec. 2008.

[3GPP 32.821] 3GPP TR 32.821 V9.0.0, Study of Self-Organizing Networks (SON) related Operations, Administration and Maintenance (OAM) for Home Node B (HNB), June 2009.

[3GPP 36.902] 3GPP TR 36.902 V9.0.0, Study of self-configuring and self-optimizing network use cases and solutions, Sept. 2009.

[3GPP 23.203] 3GPP TS 23.203 V8.4.0, Policy and charging control architecture, Dec. 2008.

Annexes

A. Additional Notes on Chemical Networking QoS And Resource Optimization

Computational Model

A molecule refers to a piece of information, which maybe code, data, or code and data, and which appear in a network packet (molecule). The adopted symbolism for a fraglet is the following:

$$id[s_1\, s_2 \ldots s_n]m$$

where id refers to the execution point in the network, m is the fraglet instance counter, reflecting its concentration in the multiset, and s_i is a string representing code instructions or data. Some of the instructions invoke a transformation (packet re-writing), while others when another suitable molecule is present in the soup (determined by matching tags in position s_2) result in a reaction according to the rule-set R. The combination of transformation and reaction operators (instructions), enable touring complete programmability. A non exhaustive list of valid transformations and reactions is given in the following table.

Reaction rules	
Transformations	
[***dup*** tag sym tail]	→ [tag sym sym tail]
[***exch*** tag sym1 sym2 tail]	→ [tag sym2 sym1 tail]
[***split*** head * tail]	→ [head] [tail]
[***nul*** tail]	→ []
id$_1$[*send* chnl id$_2$ tail]	→ id$_2$[tail]
[*nop* tail]	→ [tail]
[*wait* tail]	→ wait time t then .... [tail]
[*pop* tag sym tail]	→ [tag tail]
Catalytic Reactions	
[*match* tag tail1] + [tag tail2]	→ [tail1 tail2]
[*matchp* tag tail1] + [tag tail2]	→ [*matchp* tag tail1] + [tail1 tail2]
[*matchs* tag1 tag2 tail1] + [tag1 tag2 tail2]	→ [tag1 tag2 tail2] + [tail1 tail2]

Implicit Adaptation using Chemistry

The simple example of the confirmed delivery protocol presented in Section 7.4 provides a basic flow-control functionality if the sender waits for the [ack] molecule (by picking it up with a [match ack …]). For a more complex example we implemented a flow control protocol where the sender can send up to N packets before blocking on the ack for the first message. An additional complexity is that we permit the channel to reorder (but not lose) messages, so the protocol has to re-establish the correct sequence of packets before delivering them. In this version the driving element of the system is the producer loop which injects a new data item to ship, then waits for a local 'OK' signal, after which it starts producing the next item. For each new datum we acquire a token which will identify the datum and as soon as there is such a token available, the 'OK' signal is dispatched. Thus, the producer can generate new items as long as there is a token available. The token together with its corresponding data item is then sent as a combined molecule to the remote node. Because of the reordering channel the incoming fraglets have to wait until it is their turn to proceed. This is achieved by having a waiting fraglet that matches exactly the next expected token's name. Once the next valid token has arrived and been identified we split the thread of control. One side continues towards the consumer which consumes the datum and signals that it is OK to release the token. The other thread of control is the token that waits for the consumer's signal. Once both have joined in a reaction the token returns to the originating host. There it queues in the right order and waits for its turn to start a new journey. The corresponding chemical network for the

algorithm is shown in Figure A-1 and full system can be programmed with less than 40 molecules (table below). However, without further annotations the graph looks like a complex web of chemical pathways. An interpretation of this web can be done that associates loops with the abstract entities of classical protocol implementations. For example, the producer and consumer loops (magenta) are easily detectable. Other control loops are the sender credit book keeping at the sender side and the reordering logic at the receiver side (red). Finally, we have the credit token loops (green) and the payload deliver path that crosses the protocol's metabolism (brown). The uncolored edges are either synchronization events between control loops or parameter passing operations during computations on other arguments.

What is important to realize is that adaptation in this seemingly complex web of reactions is not a programmed feature but rather emergent through the inherent dynamic operation of the system in the rate that reactions take place, as well as the presence and concentration of molecules in the nodes, optimising in this way the use of resources in the system, the traffic generated through the flows in the network and the production/consumption of data.

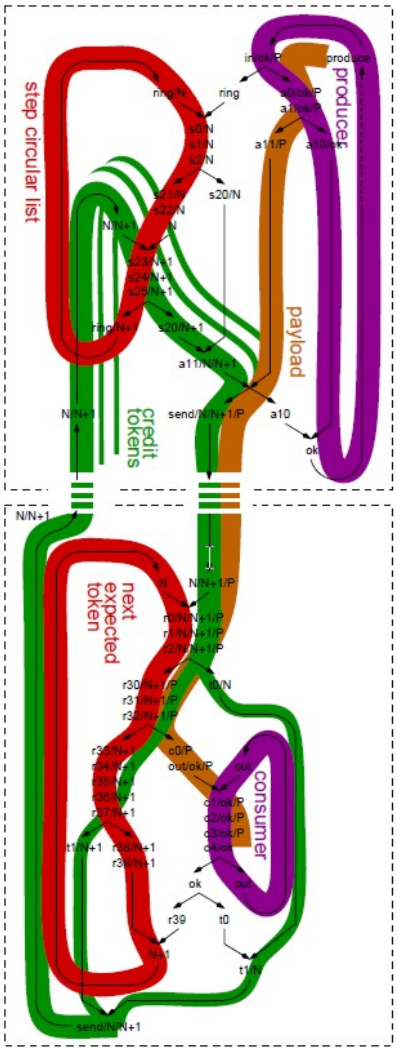


Figure A-1

Producer (Node A)	Consumer (Node B)
A[**matchp** p_produce **new** p_request] A[**matchp** p_request in ok] A[**matchp** ok **split** p_produce * **null**] A[p_produce]	B[**matchp** n0 r0 n0]
A[ring n0] A[n0 n1] A[n1 n2] A[n2 n0]	B[**matchp** r0 **exch** r1 r30] B[**matchp** r1 **exch** r2 *] B[**matchp** r2 **split** t0] B[**matchp** r30 **exch** r31 c0] B[**matchp** r31 **exch** r32 *] B[**matchp** r32 **split** r33] B[**matchp** c0 out ok]
A[**matchp** in **split match** ring **dup** s0 * **exch** a0 a11] A[**matchp** a0 **exch** a1 *] A[**matchp** a1 **split match** a10] A[**matchp** s0 **exch** s1 s21] A[**matchp** s1 **exch** s2 *] A[**matchp** s2 **split** s20]	B[**matchp** r33 **dup** r34] B[**matchp** r34 **exch** r35] B[**matchp** r35 **exch** r36 t1] B[**matchp** r36 **exch** r37 *] B[**matchp** r37 **split dup** r38] B[**matchp** r38 **exch** r39 r0]
A[**matchp** s21 **matchs** s22 match s22 **dup** s23] A[s22 **dup** s23] A[**matchp** s23 **exch** s24 s26] A[**matchp** s24 **exch** s25 *] A[**matchp** s25 **split** ring] A[**matchp** s20 **match** s26 **match** a11 **split** a10 * **send** B]	B[**matchp** out **exch** c1 **nul**] B[**matchp** c1 **exch** c2 *] B[**matchp** c2 **split**] B[**matchp** ok **split match** r39 **match** * **match** t0 **match** t1 **send** A]

Protocol design using chemistry versus traditional designs

Recently, gossip-based or epidemic protocols have gained attention because of their potential to disseminate information in a robust way. For example, the *Push-Sum* protocol in [Kem03] averages out locally stored values by means of a simple local algorithm whereby a Node i stores sum and weight of as tuple (s_i, w_i), starting with $(x_{i0}, 1)$ where x_{i0} is the node's initial (sensor) value. In each round, i.e. after a fixed time interval, each node first sends the tuple $\left(\frac{1}{2}s_i, \frac{1}{2}w_i\right)$ to a randomly chosen neighbor and to itself and collects the tuples $\{(s_m, w_m), \ldots\}$ received from neighbors in this round. Then it sums up the received values $(s_i = \sum s_m)$ and $(w_i = \sum w_m)$. In this way, the fraction $x_i = \frac{s_i}{w_i}$ asymptotically approaches the average over all values of the network. Although the protocol is simple, the "proof of the approximation guarantee is non-trivial" [Kem 03].

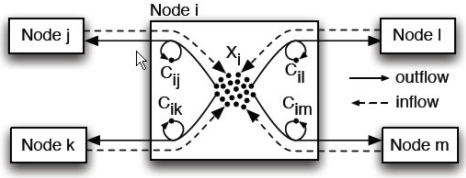


Figure A-2

An alternative, chemical implementation that is elegant and intuitively graspable. It also greatly simplifies the analysis such that the convergence proof is kept on less than half a page. We use the usual network representation as undirected graph $G = (V, E)$ where V is the set of nodes (vertices), and E is the set of edges (links). $A = [a_{ik}]$ is the adjacency matrix of the graph where $a_{ik} = a_{ki} = 1$ if $i, k \in E$, or $a_{ik} = 0$ otherwise.

Figure A-2 schematically depicts the components of the chemical *Disperser* protocol. Each node in V contains a multiset of the following molecule types: The concentration of X_i which represents the computed average and which is initially set to the local value x_{i0}. For each link $i, k \in E$ to i's neighbors there is a single instance of molecule C_{ik} that reacts with the local X_i molecule and, by doing so, sends the X_i to the corresponding neighbor node k. Formally, such a reaction is represented by the following chemical equation:

$$C_{ik} + X_i \rightarrow C_{ik} + X_k, \quad \forall i,k \in E$$

Intuitively the global reaction network should lead to equilibrium as the more neighbors a node has, the greater is the outflow of X, since there are more C molecules to react with. On the other hand, a node with higher degree also receives X molecules from more neighbors. We first write down the differential equation for the concentration of X_i:

$$\dot{x}_i = \sum_{k \in V} a_{ki} x_k c_{ki} - \sum_{k \in V} a_{ik} x_i c_{ik}, \quad \forall i \in V \tag{1}$$

where the first sum expresses the inflow of molecules in the node and the second expresses the outflow of molecules. To find a fixpoint we set $\dot{X}_i = 0$. Since there is only one control molecule C_{ik} per link we simplify the aforementioned equatin by setting $c_{ik} = a_{ik}$. Solving this equation with regard to x_i yields the following

$$\dot{x}_i = \frac{\sum_{k \in V} a_{ki} x_k}{\deg(i)}, \quad \forall i \in V \tag{2}$$

Hence, x_i is equal to the average concentration of X in i's neighbors, which only holds if the following is true

$$x_i = \frac{\sum_{k \in V} x_k}{|V|} = \langle x \rangle, \quad \forall i \in V \tag{3}$$

Consequently, at chemical equilibrium the X molecules are equally distributed over the network. This equilibrium is stable if the system returns back to the fixpoint after a small perturbation. For the corresponding analysis, we calculate the $|V| \times |V|$ Jacobian matrix of (1):

$$J = [j_{ik}] = \left[\frac{\partial \dot{x}_i}{\partial x_k}\right] = -L(G) \tag{4}$$

where $L(G)$ is the Laplacian of the following network graph

$$L(G) = [l_{ik}] = \begin{cases} \deg(i) & \text{if } i = k \\ -a_{ik} & \text{otherwise} \end{cases} \tag{5}$$

Since any Laplacian has positive real eigenvalues, the eigenvalues of (4) are negative and the fixpoint in (3) is asymptotically stable for arbitrary network topologies.

Both, *Push-Sum* as well as our *Disperser* protocols rely on a kind of mass conservation. In *Push-Sum*, half of a node's sum s is sent, the remainder is kept, however the overall sum remains constant. For *Disperser*, this conservation is obvious as the number of X molecules is conserved by all reactions. The two protocols differ in how they asymptotically approach the equilibrium: while *Push-Sum*'s code is executed isochronously, moving half of the value to a neighbor, *Disperser* transfers only one molecule per reaction, this rate being controlled by the inter-reaction time interval, which is inversely proportional to the concentration. Note that neither of the two protocols is robust against the deletion of messages. However, while a lost message in *Push-Sum* results in the loss of half of a node's value, a packet loss in *Disperser* only decreases the value by one.

Chemical Networking in the INM Framework

The INM framework in 4WARD emphasizes on the design of management functions in an integrated way with service functionality. To this extend Protocol design in chemical networking adheres and promotes this model by suggesting the coexistence of network management operations and data services inside traffic flows. At the same time (and in accordance with the broader in-network management philosophy) it promotes self-adaptation, distributed management and functional evolution in a ubiquitous way whereby data services are integrated with management processes and the environment they are delivered over. Self-adaptation in chemical network protocols is instantiated in the following properties.

Self-Healing: Program code must be self-referential and capable of regenerating itself. In addition to these structural and behavioral properties, the code dynamics must be steered by a control loop that recognizes defects and triggers code self-repair. In the chemical programming model presented so far there is no distinction between code and data. Molecules may contain code or data or both, and therefore it is possible to modify code or generate it on the fly. An example that illustrates this concept is the following "Quine", a program that generates its own code as output [Mey08]:

[*match* x *fork nop match* x]
[x *fork nop match* x]

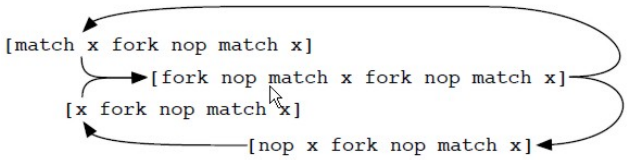


Figure A-3

These two molecules react and, by doing so, according to the rewriting rules, regenerate themselves as shown in Figure A-3.

Self-optimisation: From an execution point of view CNPs tend to exploit all available resources, leading to a competitive rather than a cooperative environment. However while exploiting resources, CNPs at the same time explore and dynamically adapt to the environment. A typical example regards the *quines*, which occupy the available system memory but survive when reducing it. Hence, CNPs achieve a high degree of embodiment whereby computation may be outsourced to the environment.

Evolution of Protocols: With a chemical protocol design, a gradual path from static to self-healing, and even towards self-optimizing and evolving protocols, could be envisioned. Mutations could be useful to evolve protocols online, potentially enabling self-optimization and long-term adaptability. We conjecture that CNPs are better suited to automatic evolution than traditional protocols due to their inherent property of multi-stability that lets them glide into the next equilibrium according to environmental conditions.

Following the management by objective principle (see Chapter 5) the operator of a network is the authority responsible for orchestrating the services that the network needs to be provide (indirectly setting in this way the competition rules in service provisioning) and setting the constraints regarding the resource distribution to be made available for the services to execute. In this setting envisioned chemical protocols should compete to deliver services by occupying resources and adapting to any changes thereof (that may occur due to technical problems or re-distribution of the resources).

In more practical terms INM domains define the borders that confine chemical protocol communication between networked systems, while self-managing entities (SEs) are the molecule soup containers where reactions take place. A set of interconnected reactions (reaction network), which essentially implement an adaptation algorithm exist in SEs as an INM capability (MCs). Quines being practically one example of such algorithms thus exist (as IMN MCs) in one or more SEs and are responsible for maintaining molecular concentrations. The presence of quines and other molecule concentrations is the central monitored *objective* through which the network operator is able to aggregate information about the state of services and the effects their execution has on the network. By nature quines and reaction networks in general can be distributed in nature across one or more soups (SEs) therefore

lending to implementations of any of the possible arrangements highlighted in section 7.2 (and according to the management objective).

Consequently, in a typical INM setup molecule soups provide an interface through which the network operator can extract statistical and other information about the existing concentrations and the dynamics of the chemical reactions in the soup. At the same time another interface is provided through which one may bias the reactions in the soup (implicit objective enforcement) through the reduction or increase of molecular concentrations, introduction of new or depreciation of reaction rules, or through the selection of specific dilution algorithms.

References

[Kem03] D. Kempe, A. Dobra, and J. Gehrke, “Gossip-Based Computation of Aggregate Information”. In FOCS, pages 482–491, 2003.

B. Co-Design for In-Network Management Additional Notes on Chemical Networking QoS And Resource Optimization

This annex details on some of the concepts that pertain to co-design of management and service functionality.

The concept of co-design originates from the observation that knowledge and functionality about how to manage a system is typically split between multiple roles involved in the operation and management of the target system. For instance, service designers and network operators may each not only know how to manage different aspects of a service, but also how to provide the functions that implement different parts of the overall management tasks. Co-design patterns proactively support the exploitation of synergies between such parties: they represent a set of structural blueprints of how to construct parts of a management system by combining knowledge and functionality of different parties to facilitate the reuse of existing functionality, to simplify management function design, and to increase system performance.

Co-design patterns are applicable in the context of embedded network management where management functions are co-located with service functions. Figure B-1 illustrates the concept of in-network management (INM), an approach to embedding management functions inside of network elements (network nodes).

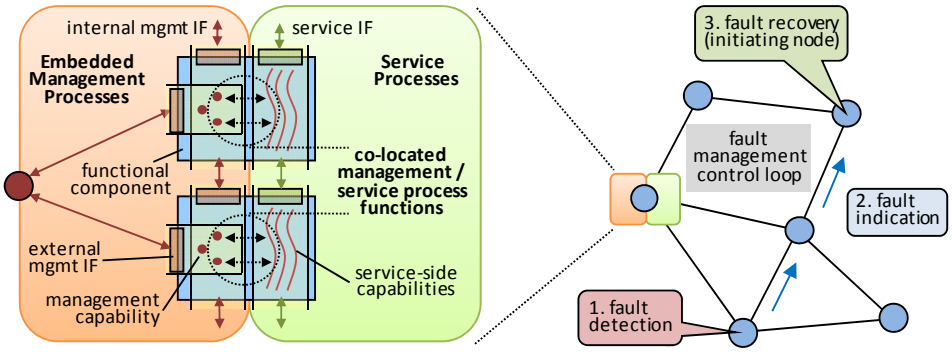


Figure B-1

Figure B-1 (left) sketches the structure of a network node, where functional components integrate both service and management logic into a single coherent, deployable entity. Management functions are invoked by calls to management capabilities, which implement algorithms that realize the management functions, such as fault handling. Typically, a management control loop is formed by multiple interacting management capabilities of functional components that span several nodes in a communication network (Figure B-1 (right)). Invocation of management capabilities by service processes and vice versa is performed, for instance, by function calls, and control between both sides is transferred accordingly. Supporting in the design of interactions between embedded management and service processes is the objective of the proposed co-design patterns.

We propose an initial nonexhaustive set of co-design patterns for embedded network management that model typical recurring problems in the fine-granular interactions between management and service functions. Three of the proposed patterns are:

Control handover: This basic co-design pattern makes explicit that control is handed between service process and management process in either direction. This pattern separates both spaces in functional terms and helps in understanding the separation of concerns in the design phase of complex management systems involving many functional components and

network elements. This pattern applies, for example, to the situation of a service-side security exception that leads to the invocation of a security-related management capability.

Informed handover: This variation of the control handover pattern uses additional information to indicate that the invoked management capability (function) is to be executed with certain constraints. Typically, constraints relate to performance and security, e.g. the maximum delay that only nodes within a network cluster must report on a management-related query. This pattern makes explicit the knowledge shared between service and management side, such that both sides can agree on this knowledge and are able to continue concurrent operation with the same assumptions. An example application of this pattern is the invocation of a management capability by a service process with specific timing requirements.

Predicate: This pattern provides defined means to evaluate a condition (predicate) on the management side where the knowledge about the condition is provided by the service side. Such situations typically occur in managing faults that can be described by service-specific properties, such as the reception of a sequence of messages that is not allowed in the case of non-faulty operation of the service. Together with a predicate, the information required to evaluate the predicate is handed to the management side for evaluation. By definition, if the predicate evaluates to false, control returns to the service process, otherwise control is resumed on the management side. Hence, the predicate pattern can be viewed as a conditional version of the control handover, and it can be combined with the informed handover. An example application of this pattern is the definition of a fault situation in the form of a predicate that is evaluated by the management side.

C. Multipath Congestion control based on Emergent Behaviour

Computer networks have experienced an explosive growth over the past decade. As a side effect of this grow, in 1986 the Internet had the first of what became a series of congestion collapses. A given network path is considered as congested if too many packets arrive at the same routers queue, resulting in an amount of packets being dropped. To avoid a congestion collapse, congestion control is essential. The basic principle of congestion control used in today's internet re-lies on a implicit or explicit collaboration of end systems and routers. It can be described as follows: In case the filling level of a routers queue reaches a pre-configured threshold, queue management strategies as e.g. RED are applied to drop packets in a random fashion. TCP connections of end-systems affected by these artificial packet losses reduce their sending rate to avoid the impending congestion. In case of UDP based communication it is up to the affected application to implement a suited flow or congestion control principle. However, the obvious ways to implement a window-based end-to-end transport protocol as TCP can result in exactly the wrong behaviour in response to network congestion as already stated by Van Jacobson in [Van88]. Especially in case of wireless links or heterogeneous environments. Further for TCP based communication it is required that both end-systems are using the same TCP variant and/or implementation for best performance. The required interaction between end-systems and routers can also lead to the unfair sharing of network resources. For UDP based transfer, an application programmer can select an aggressive flow control strategy which does not at all or only after some delay reduce the sending rate in case of packet losses. If such a strategy is combined with retransmission or forward error correction, it is possible to get a higher share of the available bandwidth when compared to end-systems using TCP. To address these issues we focus on an in-network congestion control approach which does not rely on the interaction between end-systems and routers. With the aim to reduce required configuration and management overhead to a minimum we apply an emergent phase synchronisation phenomena of pulse-coupled oscillators. The required status messages to be exchanged between routers have minimal semantics. In fact only the transmission of pulses is required, which may be transmitted on Layer 2 of the ISO/OSI Layer model or below.

Phase-Synchronisation of Pulse-Coupled Oscillators

As mention before we investigate in an in-network congestion control approach which does not rely on the interaction between end-systems and routers. With the aim to reduce required configuration and management overhead to a minimum we evaluate if and how we can apply a principle known as emergent behaviour to congestion control. In our context a system has an emergent behaviour in case each entity in the system applies simple rules (microscopic behaviour) which results in sophisticated behaviour of the overall system (macroscopic behaviour). An example for emergent behaviour is the spontaneous phase synchronisation of pulse-coupled oscillators, which is a well known phenomena in biology and physics [Pes75]. In Figure C-1 (a) we provide an abstract model of such oscillators as introduced by Mirollo and Strogatz in [Mir99].

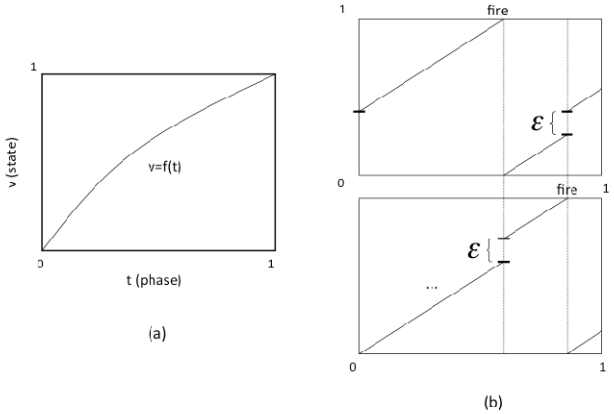


Figure C-1: Oscillator Model

In their model, a single oscillator is characterised by a concave status function f(t). For t growing from 0 to T = 1, the function f describes the charging of the oscillator up to a maximum voltage

$$V_{MAX} = 1 = f(k \cdot T) \quad k = 1, 2, 3, \ldots$$

In case an oscillator is charged up to V_{MAX} it sends a pulse of energy to all oscillators in its environment (i.e. it fires), sets its charging level to 0 and starts again the charging process. At the same time, the energy emitted increases the charging of each other oscillator by a predefined amount as shown in Figure C-1 (b). A result of the work of Mirollo & Strogatz is that each group of identical, pulse-coupled oscillators following their model reaches a state where they have synchronised their phases i.e. fire in sync (up to a set of measure zero). An application of this synchronisation property to networking problems is of interest because:

1. The observed synchronisation property is based on emergent behaviour thus no additional configuration or management is required.

2. Once archived, synchronisation corresponds to a stable equilibrium.

In fact, recent research in the networking area has investigated in the question if pulse coupled oscillators can be used e.g. for time synchronisation in Ad-Hoc networks [Tyr06][Hon05]. The main topics investigated in have been the effects of network transmission delays to the synchronisation effect, as well as the fact that in a real world environment nodes cannot send and receive sync pulses at the same time. While we can build on the obtained results we need to study further properties of oscillator groups or our purpose. In fact we will verify if a result of Ermentrout & Rinzel can be applied, which states that in case the frequency of one oscillator is (up to some extend) higher it increases the group frequency to its own [Erm84].

Sync for congestion control

To apply the sync property to congestion control we start by associating each queue in a router with an oscillator based on the Mirollo & Strogatz model. In addition we identify the filling level of the queue with the oscillator frequency. As illustrated in Figure C-2 this can be done by mapping the filling level **L** to a value v with $0 \leq v \leq \varepsilon/a$. For the actual mapping function $m : [0,1] \rightarrow [0, \varepsilon/a]$ we use the linear function $m(L) = (\varepsilon/a) \cdot L$. The value a will be defined later using simulations.

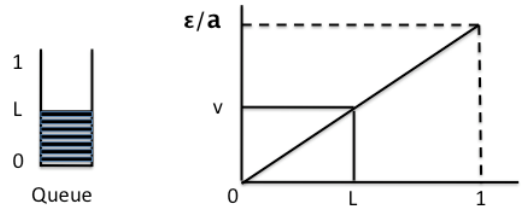


Figure C-2: Mapping of Queue Filling level to Frequency

Based on this mapping, the oscillator frequency changes depending on the Queue filling level in the range $[f^{-1}(1 - (\varepsilon/a)), 1]$. After this initial step, we describe now how the sync property can be applied to realise congestion control in a Multipath-Routing scenario illustrated in Figure C-3 (a).

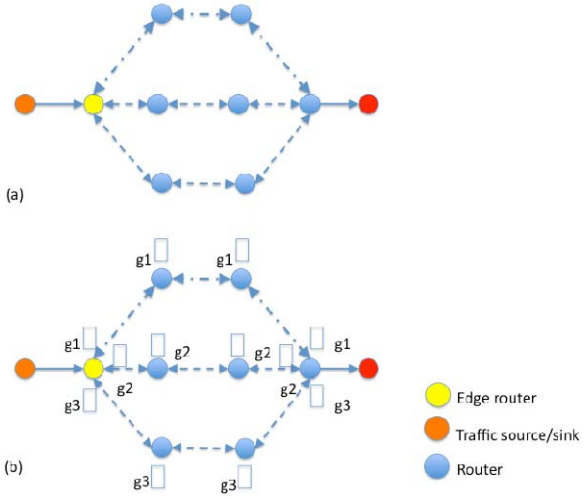


Figure C-3: Multipath-Routing Example and corresponding Oscillator Groups

In Multipath-Routing multiple alternative paths between a data source and sink are calculated which can be utilised for the actual data transfer. In our example we assume disjoint paths for simplicity. Each path calculated, defines a group of oscillators associated with the corresponding router queues (Figure C-3 (b)) which are coupled during or after path calculation.
In our example, these groups are denoted g1, g2 and g3. Based on the result of Ermentrout & Rinzel

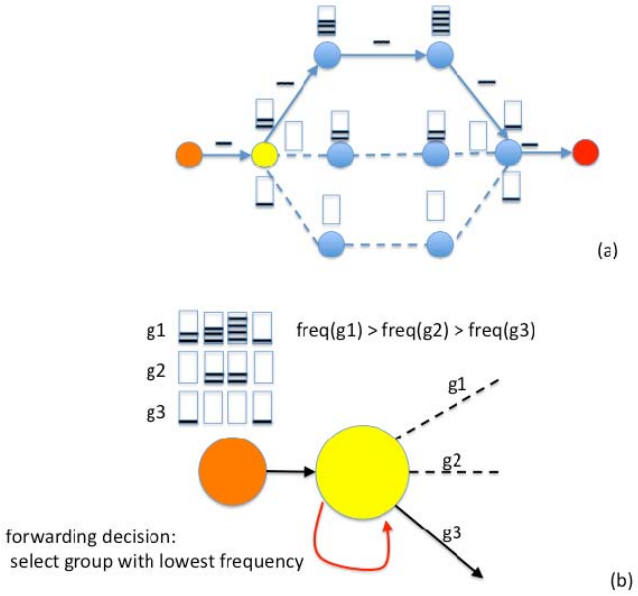


Figure C-4: Least Congested Path First

the highest oscillator frequency in g1, g2, g3 defines the frequency of the whole group. Consequently, in case of congestion as shown in Figure C-4 (a), the edge router is able to determine the path with the highest and lowest maximal filling level by comparing the frequencies of its oscillators.

More concrete by using their fire times T_{g1}, T_{g2}, T_{g3}, the before defined mapping function and the status function f (t) of the used oscillators it is possible to calculate the corresponding maximum filling levels of the queues in g1, g2, g3 using the function:

$$L_{MAX} = (1 - f(T_{gi})) \cdot (a/\varepsilon),\ i = 1, 2, 3$$

Based on this information, the edge router can always choose the Least Congested Path First (LCPF) for the next packet to be send (Figure C-4 (b)).

Before it is possible to apply above described LCPF scheme it is required to verify that the results from Mirollo & Strogatz and Ermentrout & Rinzel hold in our context as well.

In fact the network topology used to interconnect oscillators plays an important role for the sync property. In their paper Mirollo & Strogatz provided a proof for uniform coupling which translates to a fully meshed coupling topology as show in Figure C-5 (a). For the approach proposed we use chains of oscillators (Figure C-5 (b)), and it is not evident that sync still emerges in case of such an oscillator coupling topology. In addition it is required to verify the used frequency domination property in such a case as well.

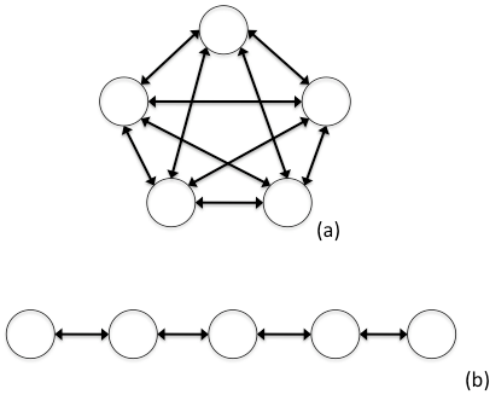


Figure C-5: Oscillator Coupling

In the following we use a simulation based approach to test the following claims:

claim 1: *After a appropriate selection of oscillator related parameters the sync property emerges also for chains of pulse coupled oscillators*.

claim 2: *For chains of oscillators the sync property also emerges in case of frequency variation. The resulting group frequency is the frequency of the fastest oscillator.*

Before we can start with the verification of these claims we first need to describe the related parameter space in more detail. As stated before each oscillator is defined by a convex status function f (t). An example for such a function given in [Pes75] is:

$$f(t) = (1/b)\ln(1 + (e^{b} - 1) \cdot t) \quad 0 \le t \le 1$$

The parameter b used controls the slope of f (t). The relation between slope and b is shown in Figure C-6.

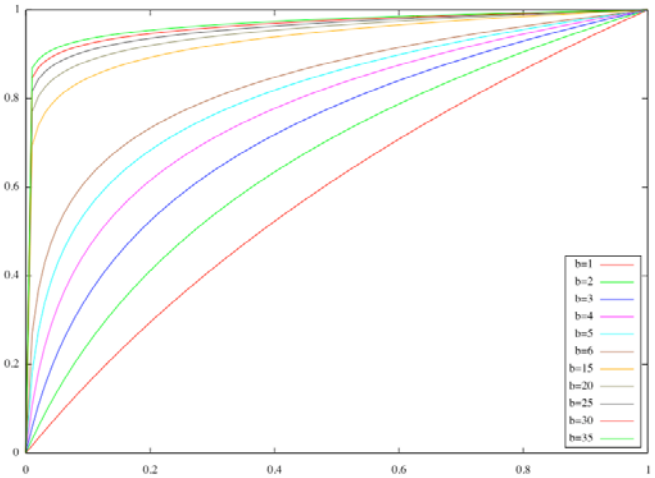


Figure C-6: Status-Function for different parameters b

Beneath the status function two further parameters are required to be defined i.e. the pulse strength and the parameter **a,** which defines the maximum frequency variation in an oscillator chain. To explore a suited parameter space for the proposed LCPF scheme we use a discrete event based simulation environment. The maximal length of an oscillator chain has been set

to 100 which would correspond to a network path with 100 routers between traffic source and sink.

To verify *claim 1* we initialised a chain of 100 oscillators with a random phase between 0 and 1. In each step of the actual simulation we shift the phase of all oscillators by 0.1 towards 1. After a shift we determine the set of oscillators about to fire. In case an oscillator is about to fire, we set its energy level to 0 and enhance the energy level of all oscillators not about to fire in this cycle by ε. For each combination of different ε and b values, we performed 100 simulation runs.

During the performed simulations we observed that for the selection of $b \geq 15$ and $0.05 \leq \varepsilon \leq 0.35$ the sync property emerges also for chains of oscillators. The obtained averaged results of the simulation are collected in Figure C-7. A general trend one can see is, that the higher the value b the faster the system approaches a synchronised state. In addition, the value = 0.15 shows best performance with regard to all b values while the combination = 0.1 and b = 30 requires the lowest amount of cycles until synchronisation.

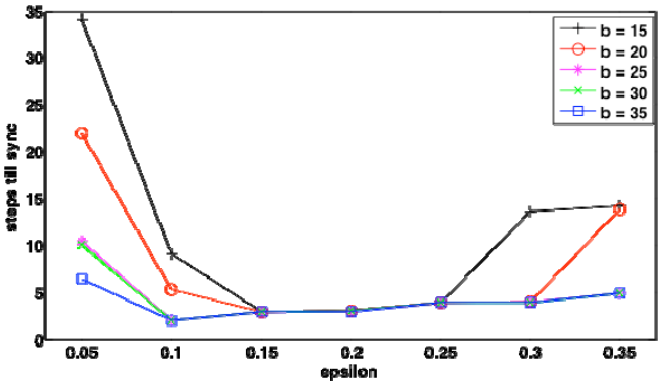


Figure C-7: Phase Variation

To verify *claim 2* we use the same simulation environment but in addition to phase variation we also variate the frequency of each oscillator. For the frequency variation we selected **a** out of the range $(1, 15) \subset R$. For ε we used the value 0.15 for all simulation runs and the phase shift has been again 0.1 units per simulation cycle. As a result of the performed simulations we can state, that for $b \geq 20$ we obtained a stable emergence of the synchronisation property. As shown in Figure C-8 only very few cycles are required to reach sync. The main reason for this effect, is the fact that for $b \geq 20$ the slope of the oscillator status function is comparable high (c.f. Figure C-6), and selection of a = 1.5 correspond already to a change of T from 1 to $f^{-1}(1 - 0.15) \approx 0.135$ for b = 20 which corresponds to a comparable high frequency.

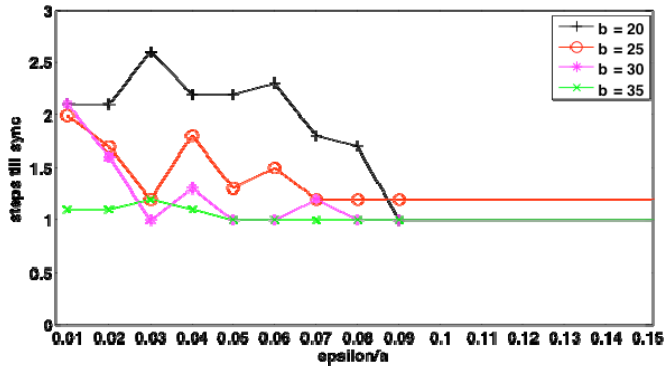


Figure C-8: Phase and Frequency Variation

To summarise the obtained results we can state, that we have been able to verify claim 1 as well as claim 2. More concrete, selecting b = 20, = 0.15 and 1 ≤ a ≤ 15 the synchronisation and frequency domination property of oscillator chains up to length 100 can be observed.

Summary

In this annex we addressed in-network congestion control based on a coupling between congestion control and routing. To reduce management and configuration complexity we put a strong emphasis on self-organisation and emergent behaviour. We described how the spontaneous phase synchronisation of pulse-coupled oscillators can be used to realise Least Congested Path First (PCPF) forwarding in Multipath-Routing scenarios. In addition we verified using simulations that the synchronisation and frequency domination property of full mesh pulse-coupled oscillators also hold in case of oscillator chains. As a further result of the simulation based study we provided a parameter set which can be used for implementation. The proposed approach has several promising properties. It is self-organising since the used synchronisation property is emergent. The described LCPF scheme is independent of interaction with End-Systems and TCP. The control messages required to be transmitted between neighbour routers in a given path have minimal semantics. In fact just a transmission of pulses is required which can be approached on Layer 2 of the ISO/OSI Layer Model or below. A realisation of the oscillators in hardware e.g. on the interface card of a router is possible as well. In the next step we address the comparison of LCPF with selected congestion control approaches including analysis of stability and reaction time to congestion.

References

[Erm84] G.B. Ermentrouta and J. Rinzel, “Beyond a pacemaker’s entrainment limit: phase walkthrough”, Am-J-Physiol. 1984 Jan; 246(1 Pt 2): R102-6.

[Hon05] S. Hong, ”A scalable synchronization protocol for large scale sensor networks and its applications”, IEEE Journal on Selected Areas in Communications, May 2005.

[Mir90] R. E. Mirollo, and S. H. Strogatz, “Synchronization of pulse-coupled biological oscillators SIAM Journal on Applied Mathematics” 50(6):1645-1662, December 1990.

[Pes75] C. S. Peskin, ”Mathematical Aspects of Heart Physiology”, Courant Institute of Mathematical Sciences, New York University, New York (1975): pp. 268-278.

[Tyr06] Tyrrell, Auer, Bettstetter, “Firefly Synchronization in Ad Hoc Networks”, MINEMA workshop, 2006.

[Van88] V. Jacobson, “Congestion avoidance and control”, ACM SIGCOMM88. Stanford, California, USA, August 1988.

D. Self-Adaptive Routing in Wireless Multi-Hop Networks

Various simulation results showed performance gaps of different protocols and metrics. However, most of those evaluations were not fair in such a way that increased performance was just mostly shown within a certain scenario [Ram03] [Awe04] [Cou03]. In the meantime there also exist some evaluations that try to close this gap, though it is still hard to interpret these results as they mostly deal with just one changing parameter [Dra04] [Bro98] [Wah08]. To overcome this lack of information, we conducted simulation experiments in various scenarios in ns-2 [NS2] and present the results in the following.

Simulation Environment

As already described the used simulation environment is ns-2. In the first step we simulated combinations of the routing protocols DSDV [Per94], AODV [RFC3561], and OLSR [RFC3626] and the metrics ETX [Cou03] and the less Hop Count (which is the standard metric at today's Internet). These routing protocols as well as metrics have been chosen due to several reasons.

- AODV and OLSR are one of the most well known wireless multi-hop routing protocols that even made it to be standardized as a RFC.
- We do not assume the knowledge of positional information, hence position based routing protocols cannot be used.
- We want to evaluate protocols with different behaviour (e.g. pro-active and reactive).
- We do not run simulations with DSR as the basic idea of DSR and AODV are pretty similar and AODV outperforms DSR in almost every survey [Bop01] [Das00] [Bro98].
- We run simulations with DSDV even though DSDV is a pro-active protocol as OLSR, as exchanges of control packets is done a different manner.
- It has been shown that metrics, such as RTT or Packet Pair suffer from self-interference and perform poorly [Dra04].
- ETX and Hop Count seem to be promising routing metrics and outperform each other depending on the chosen mobility [Dra04].
- There is no multi-rate support for ns-2 if the propagation model is chosen as Shadowing (there will be more technical discussion later on) which makes it meaningless to use metrics such as ETT [Awe04].

Simulation Parameters

Parameters of the simulation scenarios are chosen in a way that the simulation outputs (tracefiles) give meaningful results about the suitability of the aforementioned approaches regarding following criteria: scalability, node density, congestion, and mobility patterns.
This way we simulated the aforementioned protocols and metrics in scenarios with following parameters.

A)

In the first run we chose following static parameters

- Network size of 600m x 600m
- Pause time of 0s (a node waits 0 seconds when it reaches the destination point before moving towards the next destination)

and following changing parameters:

- number of nodes 10 – 100 (increased by 2 each run)
- number of UDP flows 5 – 25 (incr. by 10 each run)
- node speed 0 m/s – 20 m/s (incr. by 5 each run)

B)
In a second run we chose following static parameters

- Network size of 1500m x 1500m
- Pause time of 0s (a node waits 0 seconds when it reaches the destination point before moving towards the next destination)

and following changing parameters:

- number of nodes 50 - 300 (incr. by 10 each run)
- number of UDP flows 5 – 45 (incr. by 10 each run)
- node speed 0 m/s – 20 m/s (incr. by 5 each run)

Another important factor that has to be taken into account is the propagation model of the simulation environment. There exist various forms of propagation models in ns-2 whereas the two most well known are called “TwoRayGround” and “Shadowing”. The former one is often used as standard and also supports transmission with different transmission speeds, also called multi-rate. However, there is an immense drawback with this approach as it doesn’t take any fading into account, meaning that each transmitted packet will be delivered with a probability of 100% until a packet is sent beyond the specified maximum transmission range, e.g. 250m, which leads to a packet drop. That means a packet is either transmitted with 100% delivery probability if it is within the specified maximum transmission range or 0% if it is out of it. The Shadowing model does not support multi-rates, though it supports the very important phenomenon of fading. Unfortunately, using a propagation model that does not take fading into account would lead to almost meaningless results at all, which implies that using Shadowing as propagation model is the way to go.

Explanations about chosen parameters for figures
Before starting to interpret results and graphs it is important to note, that all following figures have been selected among multiple others as it is not appropriate to put all figures within this annex. However, even though the chosen ones just reflect a certain network scenario, we will use combinations of different meaningful figures to conclude about the performance according to criteria such as scalability, node density or mobility.
Lastly, some words about the format of the figure are necessary to understand the selection of the respective axes and to understand the significance of the figure. Each figure is labelled with a title, which indicates the parameters that have been chosen throughout the simulation run. E.g. a title called “size 600x600, speed 15, flows 25” means that the network size of the simulation is x = 600m and y = 600m, the movement speed of each node is in between 0m/s – 15m/s, and the number of active UDP flows is 25. The x-axis of each figure represents the number of nodes and the y-axis the number of received packets at the application layer. It is worth to notice that the y-axis represents absolute numbers and not relative ones due to following reasons. As TCP suffers from bad performance in wireless environments (due to its misinterpretation of packet loss as congestion) we chose UDP as transport protocol. Contrary to TCP, UDP does not set up any connection before transmitting its data packets, which leads to the fact that applications might try to send packets even though the destination is not reachable. Hence, giving a ratio about successful data transmission, e.g. by means of sent/received is not appropriate and would require some adjustments. However, even if the amount of all sent data would be adjusted by those that could not be sent by the source node due to missing routes (adjusted sent = sent – dropped at source), it would be still not appropriate to use a ratio like (adjusted sent) / received. On the one hand, such adjusted number of sent packets might be imprecise as packets might not be dropped immediately at the routing layer, but due to full queues. On the other hands a ratio of 1, meaning all transmitted packets are received, seems to be very promising in general, but does not say anything about the amount of data that could be transmitted. E.g. an approach that just offers

a ratio of 0.5, but can transmit 100 packets is clearly preferable compared to an approach with a ratio of 1 that can just transmit 2 packets.

Simulation Results

A)
Figure D-1 shows simulation results of AODV, DSDV and OLSR with metric combinations of less Hop Count and ETX on a square of 600x600m (compare simulation parameters a).

Node density
For a first performance evaluation regarding the criteria node density we will choose the third figure exemplarily. The figure represents the simulation results of nodes with low mobility (0-5m/s) and 25 UDP simultaneous connections on a 600x600m square. For low numbers of nodes it is hard to conclude about the efficiency of the simulated approaches as there often does not exist a route between source and destination. When the node number slightly increases (around 20) it gets obvious that OLSR with ETX performs the best. However, at a node number of around 50 the performance clearly changes with favour to AODV and the Less Hop Count metric. Again, the only changing parameter in this figure is the number of nodes, representing the node density. In other words, the lower the node density the better performs OLSR and ETX and the higher the node density the better AODV and Less Hop Count. Moreover, it is noticeable that a combination protocol + metric combination of OLSR + ETX and AODV + Less Hop Count seem to be valid every time as they always outperform their corresponding combinations OLSR+ Less Hop Count and AODV + ETX.

Traffic load
To get an idea about the impact of traffic load in the network we will compare the performance of the second and third figure. These scenarios just differ in the way that the number of active UDP flows is increased from 5 to 25, meaning “low” and “high” data traffic in the network. Basically, we can confirm the results concluded before, though we can clearly state a shift in the node density which leads to a performance change from OLSR + ETX to AODV + Less Hop Count. As previously described, the third figure shows that the performance of the protocols + metrics change, if the number of nodes exceeds 50. However, in the second figure we can see this “threshold” shifted to a value of around 70, meaning that the node density can be much higher. In other words, the load of the traffic also clearly affects the performance in such a way that the lower the network load is the better performs OLSR + ETX whereas the higher the network load is the better performs AODV + Less Hop Count.

Mobility
Finally, to conclude about the impact of mobility we compare the results of all four figures, whereas the first represents a static scenario, the second and the third a scenario with low mobility and the last a highly mobile one. To also show results of a scenario with neither low nor high traffic load we choose an intermediate value of 5 and 25 UDP flows - 15 UDP flows. As the results show the mobility has a huge impact at the performance of the protocols and metrics. Let’s take the extreme cases of the static and highly mobile networks first. In case of the static scenario, the first figure, OLSR + ETX continuously performs well whereas OLSR in general outperforms any metric combination of AODV. Again we can see that the performance gap in between OLSR and AODV gets smaller if the node density grows. In case of the highly mobile scenario AODV outperforms OLSR even for low node densities. A combination of AODV + ETX seems to be preferable just for very low number of nodes. Other than that (if the number of nodes are higher than 30) AODV + Less Hop Count performs continuously the best.

Conclusion
Again, we conclude that indeed there is a clear impact of the node density, traffic load and mobility on the aforementioned protocols and metrics. According to the results a combination of OLSR + ETX seems to be most preferable if the node density, the traffic load low and the mobility are low. Contrary a protocol + metric combination of AODV and Less Hop Count

should be chosen in case of high node density, high traffic load and high mobility. Anyway, we note that DSDV always performs badly and especially a combination of DSDV + ETX should not be used in any case. Due to broadcasts of new control packets if routing information change, DSDV will continuously flood the network when using ETX as metric (remark: ETX is computed based on the amount of ETX control packets received within the last 10 seconds the ETX-value will constantly change).

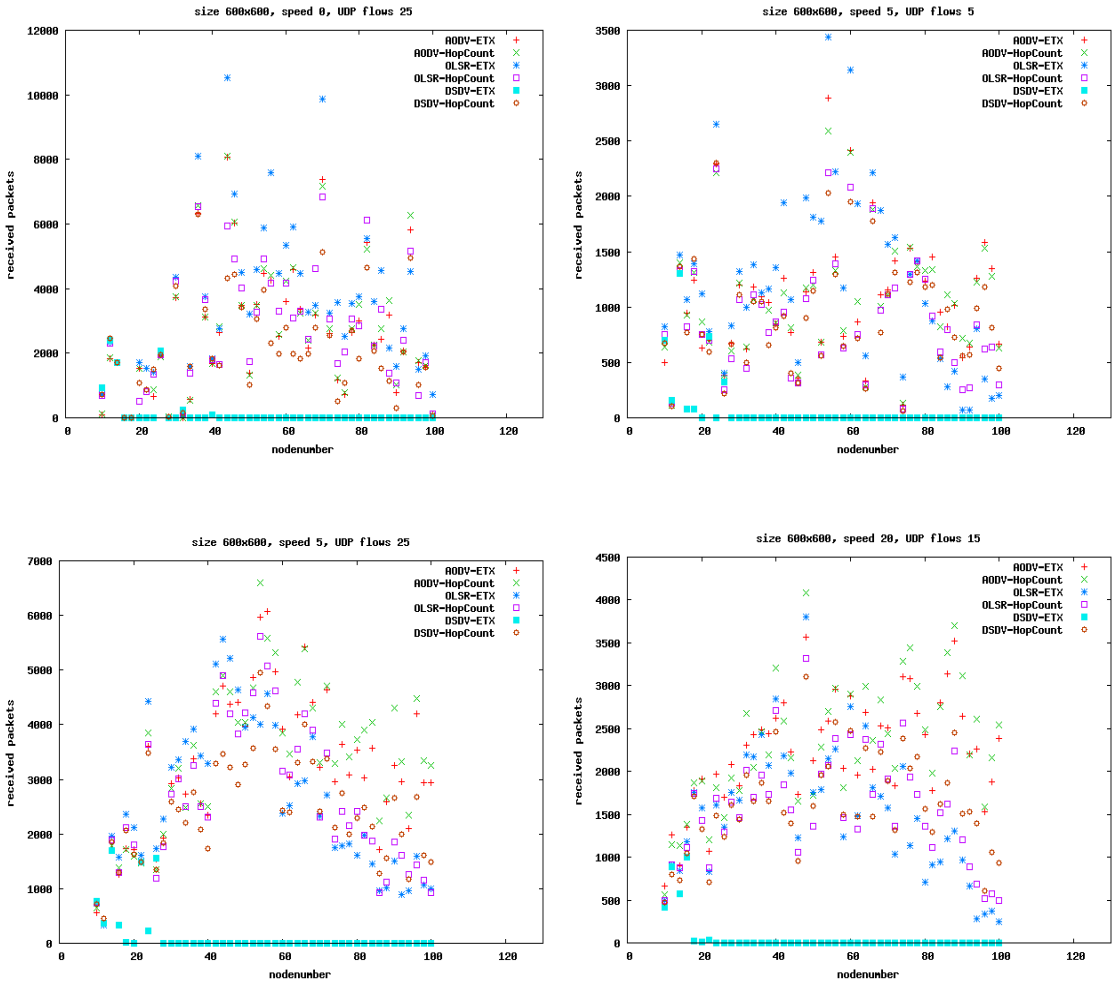


Figure D-1: Performance of AODV, OLSR, DSDV with combinations of Less Hop Count and ETX on a 600x600m square

B)

Figure D-2 shows simulation results of AODV, DSDV and OLSR with metric combinations of less Hop Count and ETX on a square of 1500x1500m (compare simulation parameters a).

Node density and scalability

Similar to results of A) we can state that for low node density (Figure D-2) OLSR with ETX clearly performs best. However, when the node density and accordingly the average number of neighbouring nodes increases the performance changes and either OLSR with Less Hop Count or AODV with Less Hop Count gets more appropriate. This result is slightly different than the one we concluded at a) where AODV with Less Hop Count clearly was the best variant. To understand the reason of this change we have to notice again that the difference

between scenario A and B is just due to an increased network size (from 600x600 to 1500 to 1500) and thus due to an increased number of nodes (from 20-100 to 100 – 300) - the node density itself is pretty constant in both cases. Again, the important factor here is the increased overall number of nodes, which apparently lead to scalability problems. Even in case of low mobility AODV seems to have some severe scalability problems whereas OLSR seems to be able to also deal with a high number of nodes due to its MPR (Multi-Point Relays) feature.
Anyway, both cases (A and B) show that using ETX as routing metric results in severe problems if the node density increases. The most likely reason for this is the continuously broadcasting of packets in order to be able to compute the ETX value. If the network density is low and accordingly there is not that much to transmit, there is no problem with computing ETX values. However, if the number of neighbouring nodes increases the "waste of bandwidth" increases as well if ETX is computed and thus seems to lead to bandwidth constraints as there is not that much bandwidth left for data transmission.

Traffic load

Analogical to A) we evaluate the effect of increased traffic load to the respective approaches. In case of low traffic load OLSR and ETX performs best, whereas in case of high traffic load AODV and Less Hop Count and OLSR and Less Hop Count performs better. The reason for this performance change are comparable to those concluded at A (traffic load) and B (node density/scalability). Seemingly, ETX is preferable if the network load is low (which also includes control data exchanged between nodes) and Less Hop Count gets better if there are some bandwidth constraints due to too much control overhead. Again, unlike results in A, there is no clear winner of either OLSR or AODV in case of high traffic load.

Mobility

Finally, we evaluate the impact of mobility. In the static scenario in most cases OLSR with ETX. Contrary with increasing mobility the performance changes and even in case of lower node density AODV with Less Hop Count outperforms the all other approaches. This result matches with the one we already concluded at A (mobility).

Conclusion

We can summarize the results presented here pretty similar to those A (Conclusion). In general, a combination of OLSR with ETX seems to be appropriate if the node density is low, the traffic load is low and the mobility low as well. Moreover, we can also state that a combination of AODV and Less Hop Count seems to be appropriate if the node density grows, the traffic load increases and in case of high mobility. However, differently from A we realize that indeed there is also an impact of scalability. If the number of nodes gets too high the metric "Less Hop Count" gets more and more attractive. Furthermore we cannot keep the conclusion that in general AODV seems to be more appropriate if the node density or traffic load increases. This is just true if the network topology itself will not change (static scenario) and thus routes will be fully stable and not change during data transmission. Other than that and if the traffic load is too high (either due to control or data packets) AODV leads to severe scalability problems and thus OLSR gets more appropriate. Finally, again we conclude that there is no case when any combination of DSDV seems to outperform the other approaches.

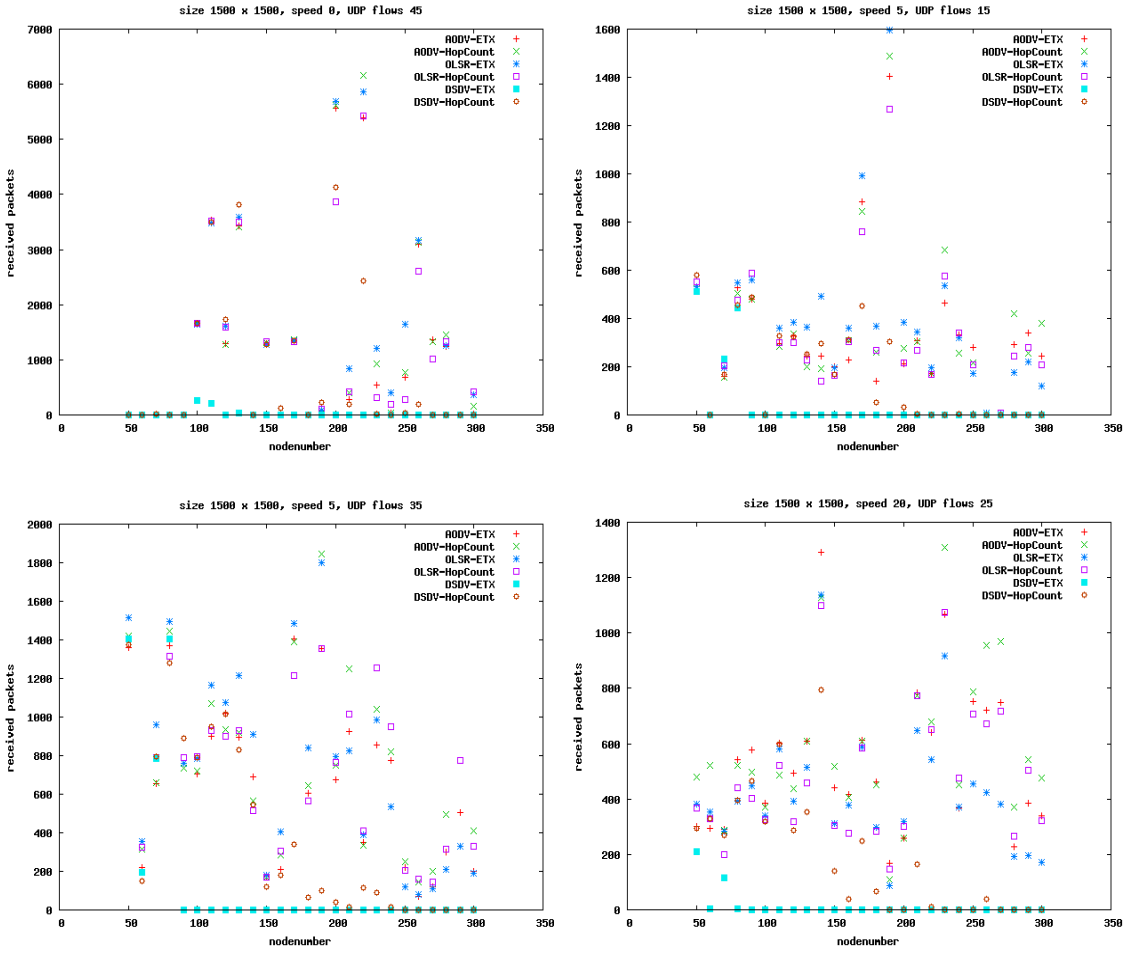


Figure D-2: Performance of AODV, OLSR, DSDV with combinations of HopCount and ETX on a 1500x1500m square

Hybrid approach with fixed zone and fixed metric

Due to our conclusions at A and B it seems obvious that using a single approach does not seem to be appropriate if there are different network conditions inside the network. Those conditions even do not mandatorily need to change during time, yet there would be a need to adjust the routing behaviour of each node. In case of changing network conditions things get even more difficult as these adjustments would be required continuously over time. Based on our evaluation we propose to combine pro-active (OLSR) as well as re-active (AODV) routing schemes with Less Hop Count as well as ETX metric.

The most well known protocol that enables the usage of a pro-active and reactive routing protocol is the Zone Routing Protocol (ZRP) [Haa97]. ZRP defines a zone which represents the number of hops (e.g. 3) to which control data (topology information) is distributed. That means that a node is aware about its z-hops neighbourhood and will use a reactive routing protocol if it no routing information to the destination is available (the destination is outside the defined zone). In order to get an idea about the performance of ZRP we simulated ZRP with some minor modifications. We simplified the intra- and inter-routing in such a way that if no routing entry is found in the routing table, the reactive protocol floods the whole network for a routing path instead of using “border nodes” at the end of each zone to flood the network. We will refer to this approach as Hybrid Routing Protocol (HRP) by using OLSR and AODV as

pro-active and reactive protocol. Simulation scenarios will be A and B again and the performance will be compared to “pure” OLSR and AODV.

Hybrid Routing Protocol (HRP)

Figure D-3 (600x600m) and Figure D-4 (1500x1500m) show simulation results of HRP with different zones, OLSR and AODV. Respectively, the first two plots represent results using ETX as metric, the latter two using Less Hop Count.

Conclusion

We won’t analyze each plot the same way according to the criteria node density, traffic load, mobility and scalability as before, but just give a more generalized statement. As the plots show in most cases (especially if they are not extreme cases) HRP outperforms OLSR and AODV showing that a combination of these approaches is indeed useful. However, it is pretty difficult to decide which zone should be used as default as the performance varies. In general, the lower the mobility, the lower the traffic load and the lower the node density the better is a higher default zone (e.g. 4-6). Contrary, the higher the mobility, node density and traffic load the more appropriate is a smaller zone (2-3). These results also match with the simulation results presented before as a small zone means a minor impact of OLSR and a major impact of AODV and vice-versa.

However, even though HRP can outperform OLSR and AODV in various scenarios, the problem remains that it is required to pre-define a fixed zone - which obviously should be adapted according to the given network condition. Moreover, we concluded before that either one of the metrics Less Hop Count or ETX is more useful according to a given network situation. Unfortunately, HRP also just uses one pre-defined metric for computing a path from source to destination.

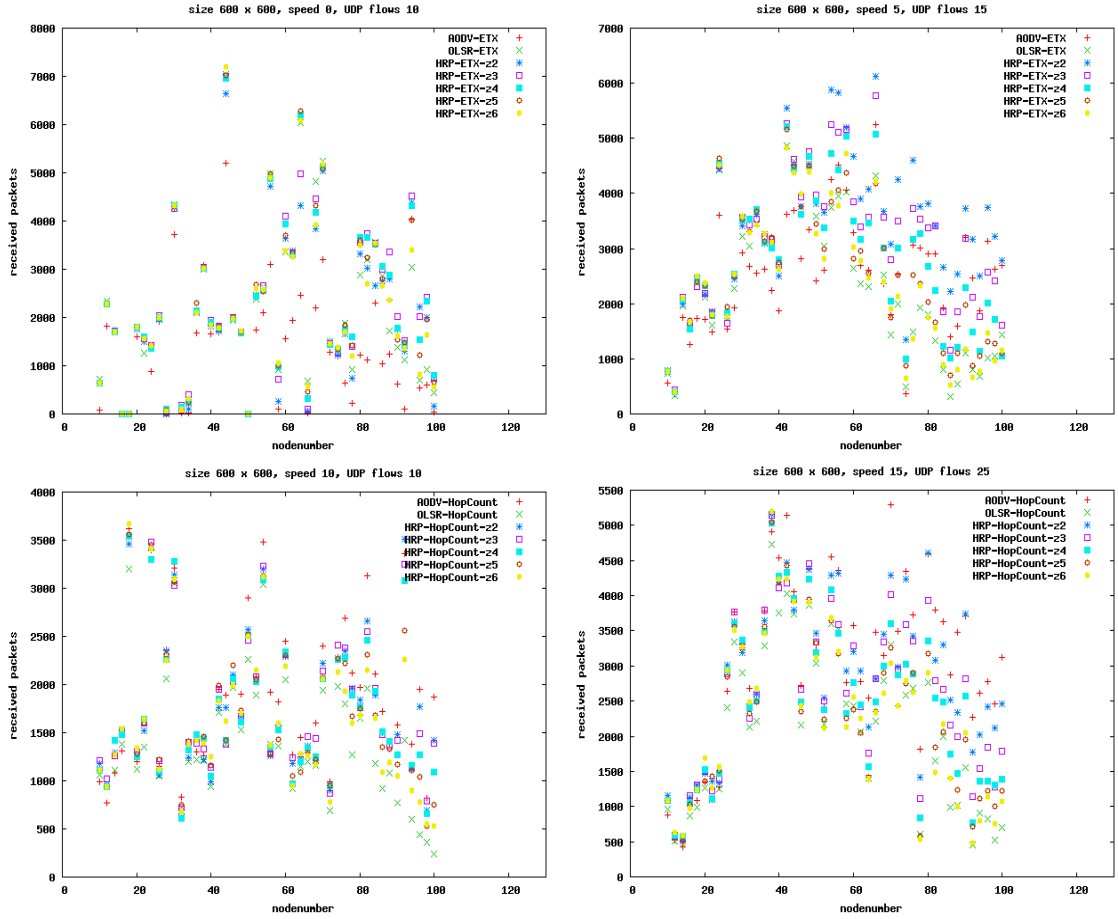


Figure D-3: Performance of AODV, OLSR and HRP with ETX on a 600x600m square

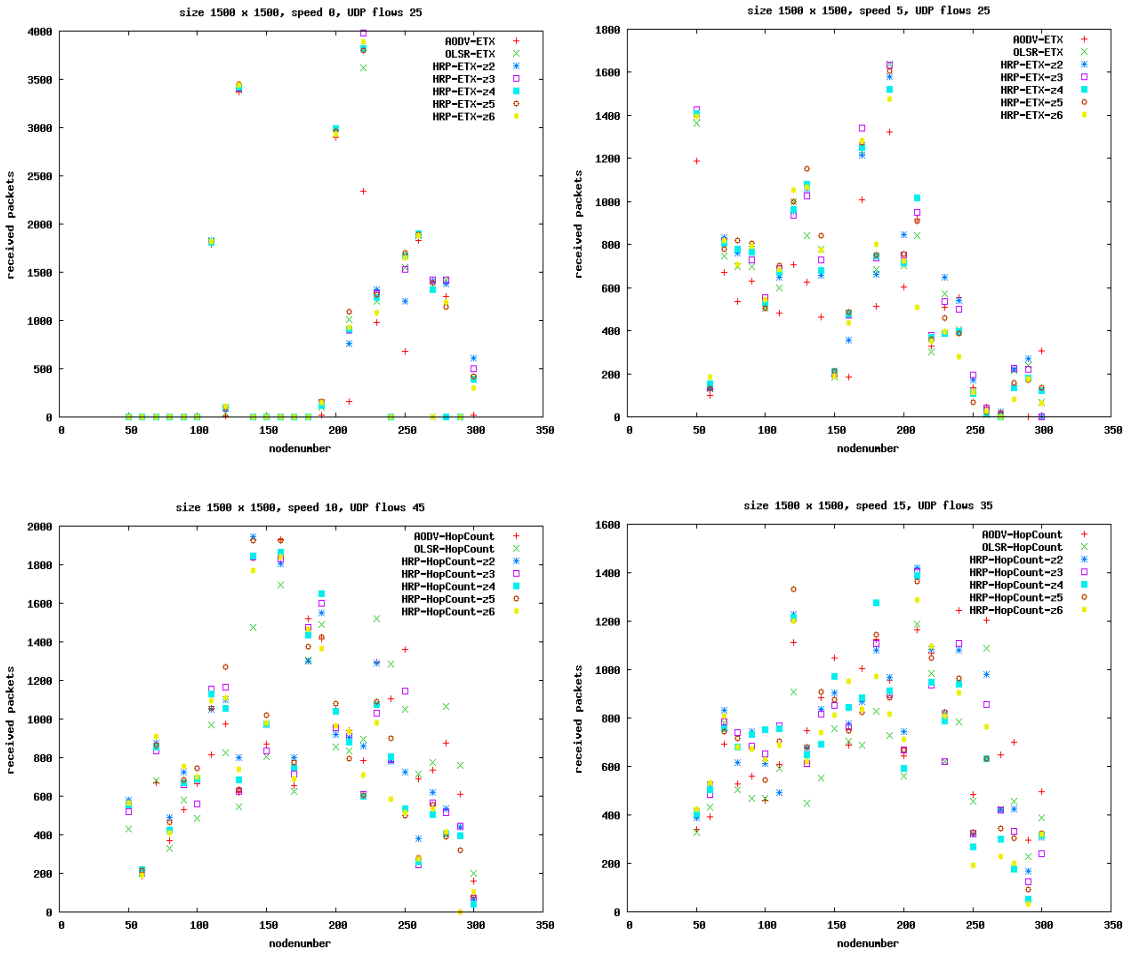


Figure D-4: Performance of AODV, OLSR and HRP with Less Hop Count on a 1500x1500m square

Adaptive Hybrid Routing Protocol (A-HRP)

To overcome this lack of adjusting parameters we propose a new approach called Adaptive Hybrid Routing Protocol (A-HRP). The idea of A-HRP is inline with the conclusions derived so far. The behaviour of the protocol is based on the decision of each participating node. Each node, regardless whether it is sending, receiving or relaying node, continuously adapts its parameters according to the network situation it experiences. That means that a data flow might get forwarded along a path based on different protocols and metrics, namely OLSR, AODV, Less Hop Count and ETX. Compared to HRP the algorithm is based on two major changes and works as follow.

Firstly, each node broadcasts periodically ETX-control packets in order to derive the link quality and to compute the ETX value afterwards. The ETX value will be used then as routing metric in combination with OLSR - thus if OLSR knows a route to a destination ETX will be used. Contrary if there is no route known to the node, it uses AODV as routing protocol in combination with the Less Hop Count metric. Accordingly, a packet will be routed based on

the Less Hop Count metric by AODV until it reaches a node that has a routing entry to the destination based on its OLSR information. From this node forth the packet will be routed via OLSR using ETX. This mechanism gives a nifty opportunity to combine the merits of OLSR and ETX with AODV and Less Hop Count. However, as criticized before there is also a need to get rid of the concept of pre-defined zones, which is the second major change.

What is different than presented before in HRP is the fact that upon receiving an OLSR control packet, A-HRP allows each node to modify the Time To Live (TTL) of this packet. This way control information can be distributed completely asymmetric within a cloud of nodes, meaning that route information of this node can be distributed via multiple hops towards a certain direction, whereas it is just known to the neighbouring nodes in the other direction. Again, the algorithm is based on the idea that each node solely decides on its own derived situation how it deals with this information. We will use Figure D-5 as an example to describe the way how this TTL modification will be realized. Please note that even though this example is very constructed and its description how the topology information is distributed is not fully inline with the specification of OLSR, it clearly highlights the impact of this TTL reduction.

Figure D-5 gives us a snapshot of a network topology assuming that all nodes are static except node M, which is highly mobile. Given the fact that S wants to transmit data to node D, theoretically there exist two routes either S-M-E-D or S-A-B-C-D. However, as node M is highly mobile and the route probably would just last for a couple of seconds it would not be useful to continuously distribute the topology information from S via M to E or even D. Node M should realize that and thus should avoid “flooding” the network with this control information that would just waste bandwidth and thus could avoid data transmission - that is exactly how the TTL reduction works. Let us take some example nodes and go via the process step by step. Node S is a static node with two neighbouring nodes and there is no data traffic going on. Accordingly, node S is willing to distribute its topology information with the intention to reach also reach nodes that are far away and thus is sending an OLSR control packet indicating that the TTL of this information is 6. Upon receiving this information at node M (a highly mobile node) the TTL of this information is regularly decreased by one – this decreasing is identical to HRP. However, additionally node M reduces the TTL to 1 by itself, as it will be just able to relay any information for a short period of time. It might not be appropriate to keep such information available as it probably will not be used. Accordingly this information is just relayed one more hop and thus just reach node E. Contrary, the topology information of node S via node A is distributed much farther. Upon receiving the initial topology information of node S at node A (a static node with three neighbours) the TTL is decreased by 1 as usual. In addition, before further distributing this information, node A also reduces the TTL by 1 to 4 as it realizes some data traffic being transmitted by node Z. Upon receiving this information at node B, C and D the TTL just gets decreased by 1 for each transmitted hop as the network situation of these nodes does not require further TTL reduction.

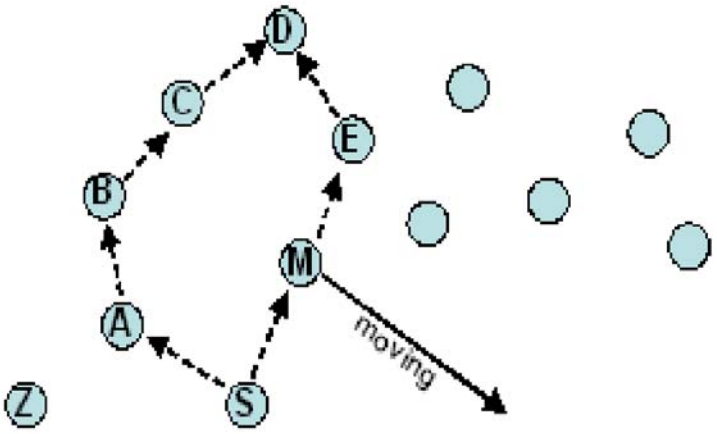


Figure D-5: Example for distributing of topology information

This example shows that OLSR routing information would be exchanged periodically between node S and D via A, B and C and a route would be already known. Moreover, it shows that there might be other routes available that theoretically even could be better or more appropriate (less hops or better quality). However, those routes could be hidden or omitted by A-HRP in order to save bandwidth for data transmission.

Based on the results achieved so far we selected a default TTL of 6 and additionally reduced the TTL by each node based on the following situation:

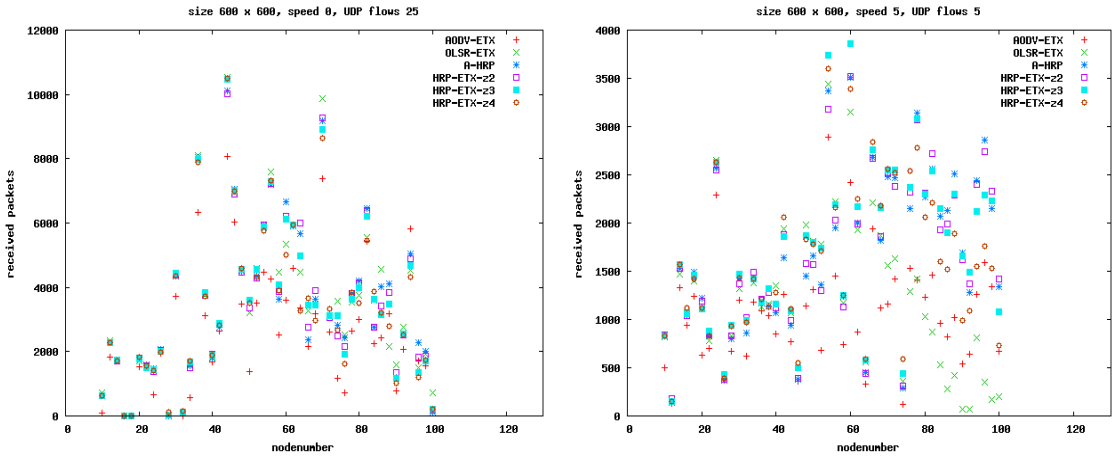
- if (neighbours >= 3) → TTL = TTL -1
- if (neighbours >= 4 && TTL >1) → TTL = TTL - 1
- if (speed >=3,5) TTL → TTL = TTL -1
- if (speed >=7) TTL → TTL = TTL -1
- if (speed >=10) TTL → TTL =1

Simulation results of A-HRP

Analogue to all previous simulations, we simulate A-HRP in the same scenarios A) and B) and compare their results with OLSR, AODV and HRP (zones 2-4). Figure D-6 shows the results for scenario A), Figure D-7 the results for scenario B). More precisely, the first half of each plots shows the amount of received packets by using OLSR, AODV and HRP with ETX as metric; the latter half with Less Hop Count. Again, there will be just shown some selected plots, though to allow a fair comparison all presented plots use the same parameters as shown before. Hence, results presented here are a summery of the ones presented before and the only alteration is the introduction of A-HRP.

A)

Firstly, we will analyze the performance of A-HRP for scenario A (Figure D-6). In comparison to all other approaches we can't realize a case where A-HRP really performs badly and is far behind all others. Contrary we can see a lot of cases where A-HRP performs best or is somewhat average. More precisely, A-HRP mostly performs better than all other approaches if the chosen scenario is not an extreme case, meaning completely static or highly mobile, or extreme low/high node density. However, we state again that even in those extreme cases A-HRP does not perform badly, but is mostly just outperformed by those approaches that have been specifically designed for this scenario.



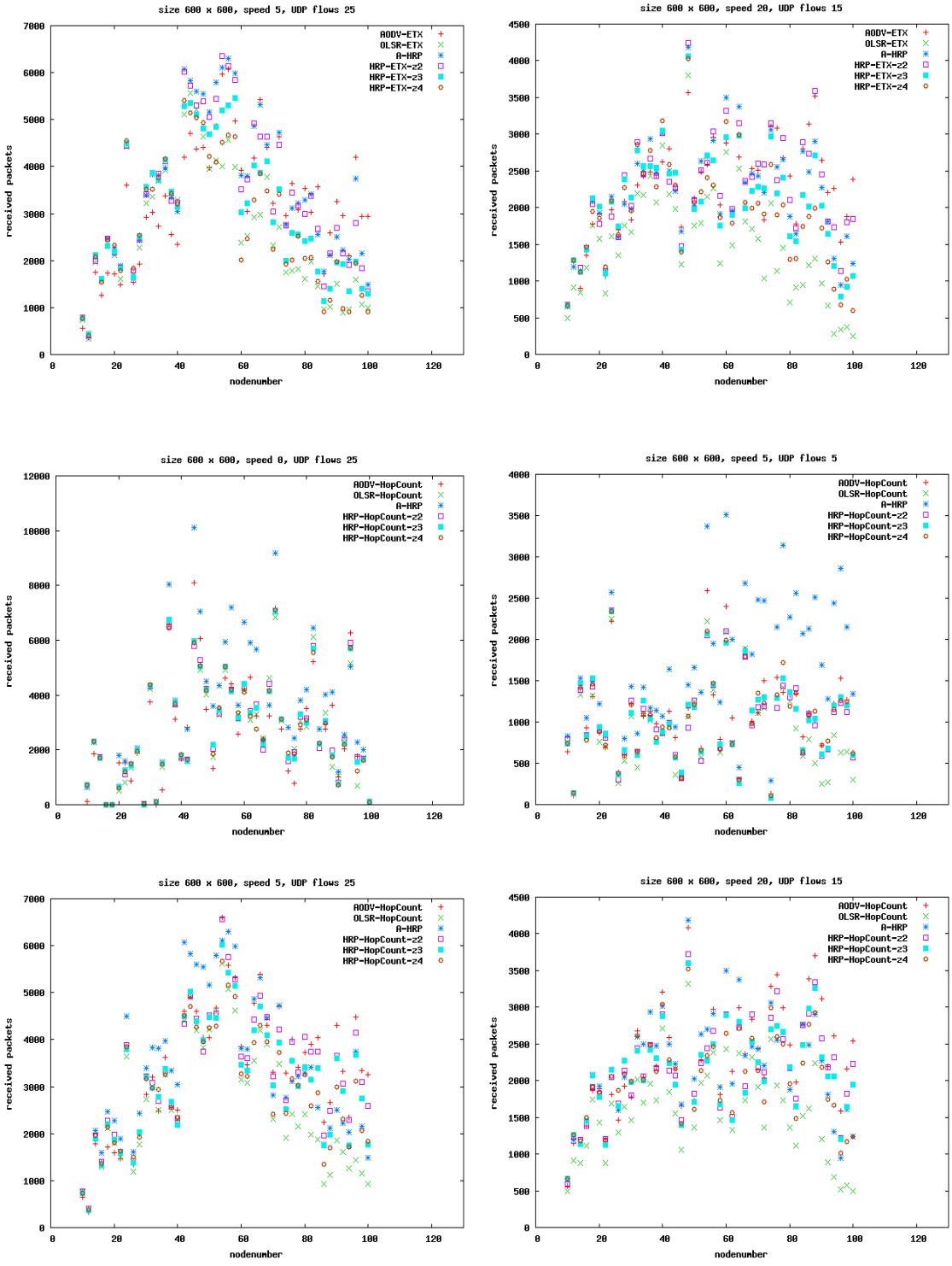
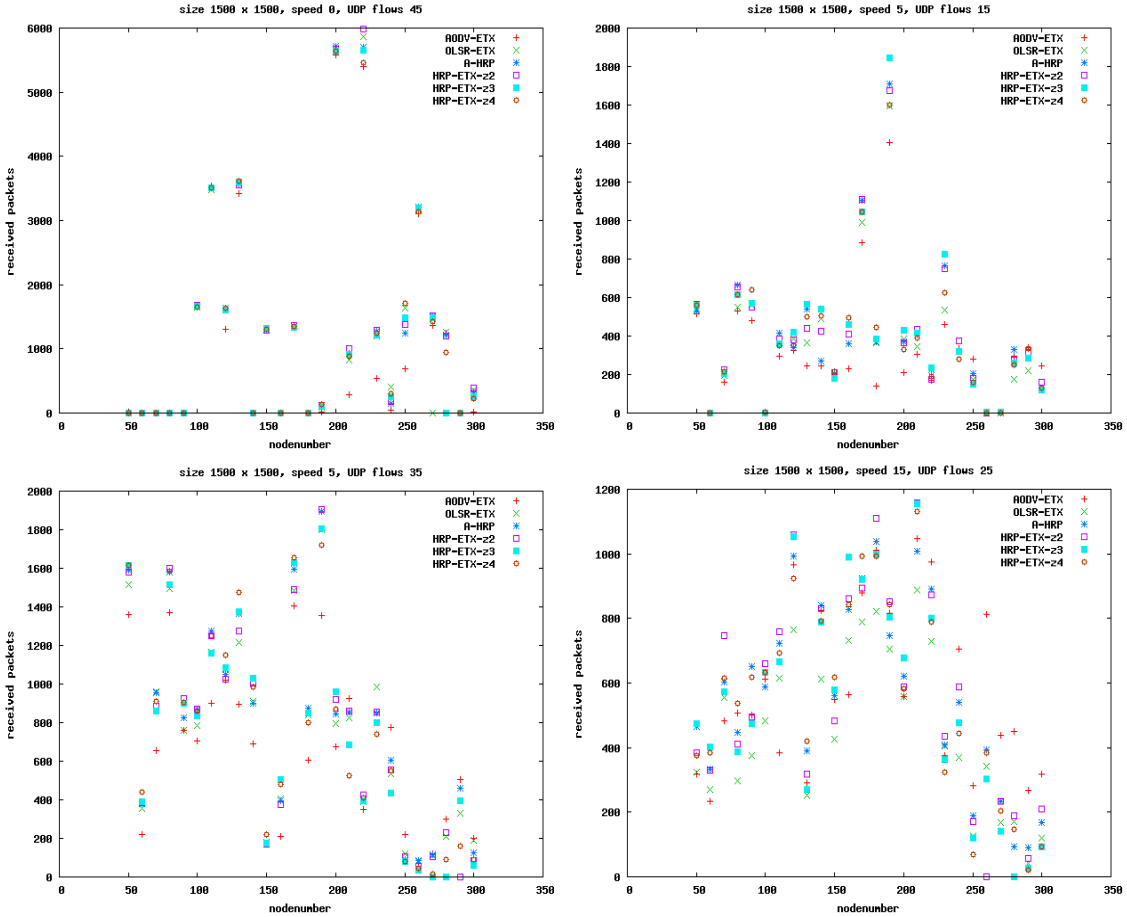


Figure D-6: Performance of A-HRP on a 600m x 600m square (according to scenario A)

B)

As described before Figure D-7 shows simulation results of A-HRP, OLSR, AODV, and HRP on a 1500m x 1500m square. Similar to the results of A-HRP for scenario A) we can see that A-HRP performs pretty well for scenarios that do not reflect an extreme case, such as highly

mobile or high density. However, differently than before we cannot keep the statement that A-HRP never performs badly. Exemplarily, we will analyze the left plot of the last row (parameters: speed 10, flows 45, Less Hop Count). In case of low node density A-HRP performs either good or somewhat about average. With increasing node density the performance of A-HRP increases as well and it starts outperforming all other approaches. Yet, when the number of node exceeds ~200 A-HRP continuously performs bad or even worse and OLSR performs best. We can identify here the same effect as already discussed before – scalability. If the overall number of nodes in the scenario is too high there emerge some severe scalability problems. Contrary, to A-HRP all other approaches are solely based on the Less Hop Count metric in this scenario and thus do not additionally flood the network with ETX control packets, which is responsible for the bandwidth constraints.



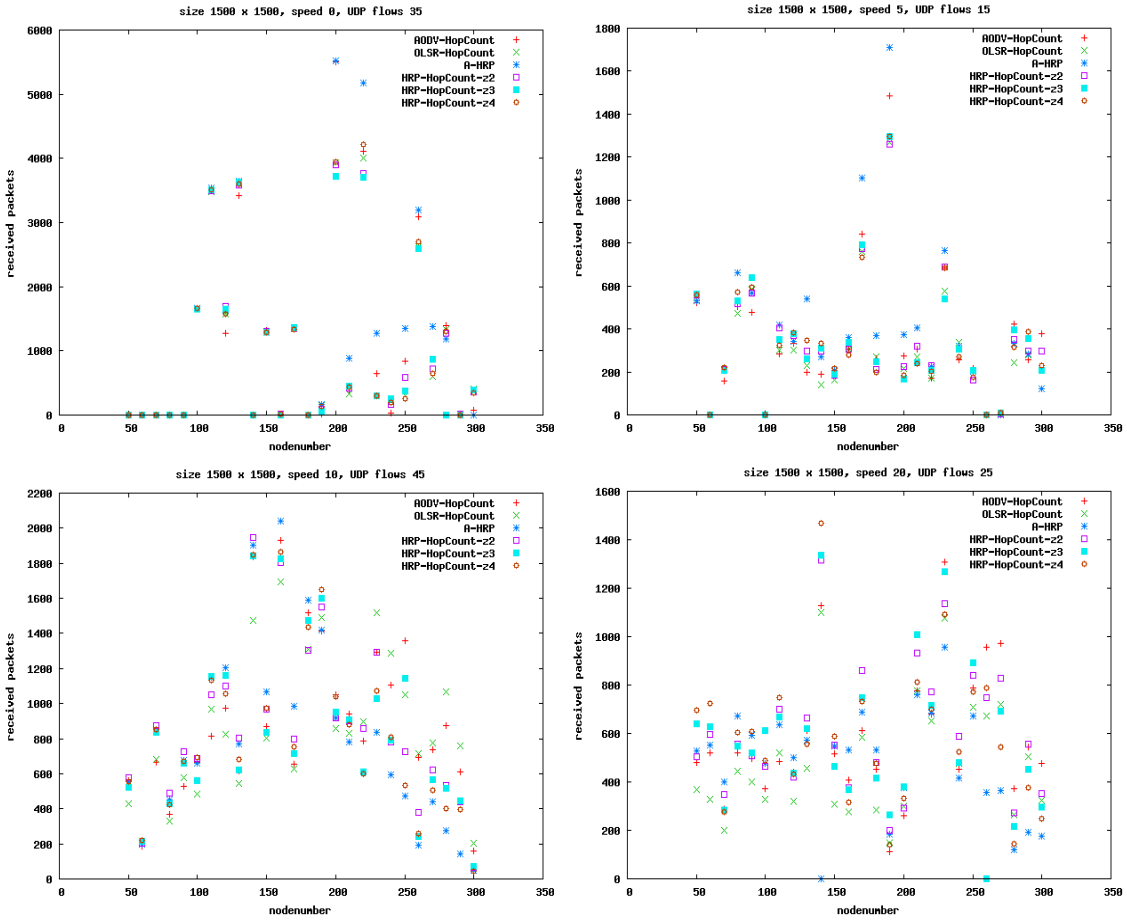


Figure D-7: Performance of A-HRP on a 1500m x 1500 square (according to scenario B)

Conclusion

A-HRP introduces two new features to existing approaches. Firstly, it combines two routing metrics that have been designed for different purposes and secondly it adaptively changes the range/zone of distributing topology information. As simulation results show A-HRP generally performs pretty well and often outperforms all other approaches. The best chosen scenario for A-HRP is a moderate scenario that does not reflect extreme cases, such as completely static or highly mobile. Hence, if the chosen scenario consists of heterogeneous devices and thus exhibits nodes with different mobility, node density and traffic load A-HRP should be used as it overcomes the lack of adapting each nodes behaviour according to the given network situation. However, if the number of nodes gets too high using ETX as routing metric results in severe scalability problems. Hence, in this case a routing protocol that is solely based on the Less Hop Count routing metric should be preferred instead, e.g. OLSR.

References

[Awe04] B. Awerbuch, D. Holmer, and H. Rubens, “High throughput route selection in multirate ad hoc wireless networks”, in Wireless On-Demand Network Systems (WONS), January 2004.

[Bop01] R.V. Boppana, S. Konduru, “An adaptive distance vector routing algorithm for mobile, ad hoc networks”, in Proceedings of the Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001), 2001.

[Bro98] J. Broch, D. A. Maltz, D.B. Johnson, Y.C. Hu, and J. Jetcheva, “A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols”, Proc. of the ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), October 1998.

[Cou03] D. De Couto, D. Aguayo, J. Bicket, R. Morris, “A High Throughput Path Metric for Multi-Hop Wireless Routing,” ACM Mobicom Conference, September 2003.

[Das00] S. R. Das, R. Castaneda, and J. Yan, “Simulation-based performance evaluation of routing protocols for mobile ad hoc networks”, MONET 5(3): 179-189 (2000).

[Dra04] R. Draves, J. Padhye, and B. Zill, “Comparison of routing metrics for static multi-hop wireless networks”, in ACM SIGCOMM, 2004.

[Has97] J. Haas, “A new routing protocol for the reconfigurable wireless networks”, Proc. of IEEE 6th International Conference on Universal Personal Communications 97.

[NS2] http://www.isi.edu/nsnam/ns/

[Per94] C.E. Perkins, and P. Bhagwat, “Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers”, SIGCOMM Comput. Commun, Rev. 24, 4, Oct. 1994.

[Ram 03] V. Ramasubramanian, Z. J. Haas, and E. G. Sirer, “SHARP: a hybrid adaptive routing protocol for mobile ad hoc networks”, Proc. ACM MOBIHOC, June 2003.

[RFC3561] Ad hoc On-Demand Distance Vector (AODV) Routing: RFC 3561.

[RFC3626] Optimized Link State Routing Protocol (OLSR): RFC 3626.

[Wah08] S. Waharte, B. Ishibashi, R. Boutaba, D. Meddour, "Performance study of wireless mesh networks routing metrics," aiccsa, pp.1100-1106, 2008 IEEE/ACS International Conference on Computer Systems and Applications, 2008.

E. Additional Notes on Change Prediction and Configuration Planning

Outside interactions

Configuration planning is done by an administrator (outside entity). Interactions take place via an API provided by the configuration module. The administrator uses the API to:

- Get security alarms displayed and detailed
- Read out the current configuration of any node at any given moment
- Read out the global configuration of the entire network
- Read out data from the prediction module
- Test new policies
- Run new rules to facilitate adding new nodes or joining a bigger network environment

The API is available in any node so the administrator can use it to either read/set an individual node or have a global picture of the entire system by querying the whole network.

Changes that are set up run through the configuration module which on it's turn runs a simulation of the proposed change via the prediction module.

Prediction module and computing model

The prediction module runs based on a Markov chain, where a future state of the system depends only on the current state.

Given a set of states $S = \{s_1, s_2, s_3 ... s_n\}$ it is said that the system has the Markov property if, by knowing the state S_i, the state S_{i+1} can be computed. The probability of "stepping" from one state to another is given by transition probability p_{ij} (where i is the current state and j is the future state). Knowing this probability means that the system is going to "step" into a new stable state; it is said that state i communicates with state j in this case.

The transition probability is always a real positive number:

$$p_{ij} = \Pr(X_{k+1} = j \mid X_k = i) > 0$$

The transition probability is obtained by computing the transition matrix. For this a stable start-up state is needed as an input. This state is either specified during the setup period, as the whole system is configured, or it can be read right after the system has been bootstrapped.

Current policies and rules apply as restrictions within each matrix e.g. for the given example, node's Z load has to remain constant, that is in the current state B. The transition matrix changes accordingly:

$$P = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 0 & 1 & 0 \end{pmatrix}$$

Each "live" node computes the probabilities using data received from it's neighbourhood thus the transition probabilities are local. This provides a better understanding of local conditions as having a "global view" does not always reflect all possible issues. A global view of the system is only created by the system administrator who can access all nodes. This reduces the computing effort in each node and also minimizes the security risks.

As the system grows bigger and more and more rules are applied also the matrices grow in size which in turn provides for a better resolution of the computed results.

As a model the Markov chain is a state machine. Transitions from one state to another happen with the “transition probability”. Each transition (step) is defined by the transition probability.

The prediction module constructs itself such a state machine. This is a dynamic process initiated by the configuration model. A given state machine is only valid for the moment it was generated at. Populating the state machine with information is done by the prediction module by requesting data from the INM. This data is arranged in the transition matrices based on existing rules and policies. After the module computes all necessary probabilities it evaluates the results and reports any findings. During these operation global rules such as system stability and security features are taken automatically into account.

Extensions

As the prediction and planning module work together for checking the new settings to be applied a possible extension for the given model is it’s ability to shape the system such that the next sate will be achieved. This can be done by specifying the next state as a target and applying the changes proposed by the prediction module as a means to achieving the result.
This method also uses few resources and may be desired in case an administrator wishes to check what changes would fit the current system.