CASE STUDY – PART I

Networks Stability and Performance

Abstract

Today networks are becoming more and more ubiquitous and dynamic. In the near future they will be able to hook together a sheer number of heterogeneous (processing, communication and storage) real/virtual resources. One of the major challenges of will be managing such large amount of resources in a cost effective and timely way in order to meet technical and business requirements, which, in turn, may change dynamically. Introducing self-organization capabilities in network control and management is seen as a potential answer to face these challenges whilst ensuring network adaptability to changing conditions. In essence, these self-organisation capabilities can be practically achieved through the exploitations of "constrained optimizations" (CO) solvers (i.e. through protocols, control loops, algorithms…spread across network layers). These solvers have to be properly orchestrated to avoid instabilities due to unwanted couplings or interactions. Actually, it should be noted that the potential occurrence of network instability might have primary effects, such as jeopardizing dramatically the performance and compromising an optimized use of resources. Ensuring stability of a strategic asset like a network of ICT resources is of paramount importance for the information society. The main goals of the case study on Networks Stability and Performance are to report on the development and demonstration of methodologies and practical approaches to detect and control the potential occurrence of instabilities in diverse network contexts. This document presents a brief description of the use case, its methods, concepts and expected innovation. The specific functional, non-functional requirements and the associated problems were presented in the deliverable D4.1 [20]. The prioritization of the problems and functional requirements were presented in deliverable D4.2 [21].

Date of release

17/09/2012

SEVENTH FRAMEWORK PROGRAMME

FUTURE NETWORKS

CONTENT

STORY LINE

The risks of instabilities are already experiences in today's networks (and also cloud computing infrastructures).

Ensuring stability of a strategic asset like a network of ICT resources is of paramount importance for society. In fact, occurrence of instability in a network may have primary effects both jeopardizing dramatically the performance and compromising an optimized use of resources.

The potential instability of an end-to-end path is a cross-layer issue: in fact, it might depend on the unwanted combination of diverse control and optimization mechanisms acting on either the underlying transport network or on the higher layers' components (e.g. flow admission control, TCP congestion control and dynamic routing).

As an example, it has been widely demonstrated that even with resource over-provisioning; a network without an efficient flow admission control has instability regions that can even lead to congestion collapse in certain configurations. Congestion control mechanisms represent another example. Currently available mechanisms (e.g. TCP Reno and Vegas) are examples of large distributed control loops designed to ensure stable congestion control of resources. On the other hand, these mechanisms are expected to be ill suited, from a stability viewpoint, for future dynamic networks where transients and capacity will potentially be much larger. A further example is the instability risk typical of any dynamically adaptive routing system: this can be (informally) defined as the quick change of network reachability and topology information, has a number of possible origins, including problems with connections, router failures, high levels of congestion, software configuration errors, transient physical and data link problems, and software bugs.

Prior art analysis about network stability issues offers other interesting examples. In [1], for example, starting from a simple model of a traditional network traffic dynamics, it is shown that a phase transition point appears, separating the low-traffic phase (with no congestion) from the congestion phase as the packet creation rate increases. In [2], an enhanced model has exhibited nontrivial scaling properties close to the critical stability point, which reproduce some of the observed real Internet features. In [3] they discuss the possibility of phase transitions and meta-stability in various types of complex communication networks as well as the implication of these phenomena for network performance evaluation and control. Specific cases include connection-oriented networks with dynamic routing, TCP/IP networks under random flow arrivals/departures, and multiservice wireless cellular networks. Wang et al. [4] presents an investigation of the dynamics of traffic over scale-free networks: results have indicated the existence of the bi-stable state in traffic dynamics; specifically, the capacity of the network has been quantified by the phase transition from a free flow state to a congestion state. Interestingly [5] addresses the risk of instabilities in Cloud Computing infrastructures. That study points out some analogies of Cloud Computing infrastructures and complex systems and elaborates on the emergence of instabilities due to the unwanted coupling of several reactive mechanisms.

Considering the evolution of ICT networks, it is widely recognized that they will become more and more ubiquitous and dynamic, hooking together a sheer number of heterogeneous real/virtual resources. So, one of the major challenges will be managing said resources in a cost effective way in order to meet technical and business requirements whilst ensuring stability and performance. Development and exploitation of self-organization capabilities in network management (and even into nodes) appears to be an answer to face such challenges in spite of the ever growing complexity of networks.

Practically, network self-organising capabilities can be achieved through the exploitations of "constrained optimizations" (CO) solvers (i.e. through protocols, control loops, methods[1], algorithms…) spread across network layers. This is true also for today Internet. On the other hand these CO solvers have to be properly orchestrated to avoid unwanted couplings or interactions, potentially bringing to non-linearity in the network behaviour.

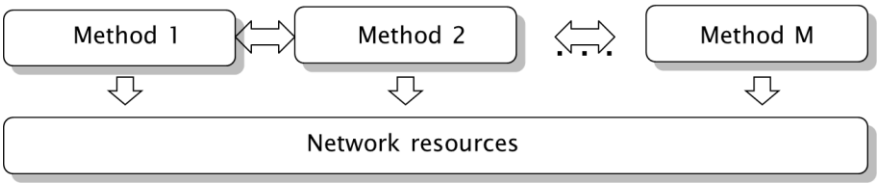

Fig. 1. Unintended coupling or interactions of multiple methods may create instabilities.

[1] Method is a general procedure for solving a network problem.

Context 1: Stability issues for Congestion Control

In this section three contexts that have been elaborated as reference examples from which are starting UC2 investigations about stability.

Overall a network can be modelled as the interconnection of resources/links carrying the data generated by users/sources. Associated with each source is a route, which is the collection of links through which information from that source is flowing. In tradition congestion control, each link sets a price per unit of flow, based on the aggregate flow crossing that link, and the sources set their transmission rates based on the aggregate price they detect. In the absence of delays, this scheme is globally stable. In the presence of delays (or large bandwidth) the scheme can become unstable. Sources try maximizing individual profit based on their own utility function, on the other hand, links uses prices to align, exactly or approximately, sources "selfish" behaviour. As such, congestion control mechanisms exploited in Internet represent one of the largest deployed control feedback system, or CO solvers.

On the other hand, currently available controllers for resource allocation (e.g. congestion control mechanisms like TCP are example of large distributed control loops) are deriving the state for a desired equilibrium point and they don't take account of transient behaviours typical of closed-loop system. TCP implicitly maximizes a sum of utilities over all the connections present in a network: this function shows the shape of the utility function for a single. But transient behaviours are not taken into account, so even if we have a globally asymptotically stable equilibrium points (corresponding to the maximization of the utility function), it is not clear how the network operates during the transients.

It's also interesting noting the instabilities that may occur when trying to use the proactive congestion control mechanism of TCP Vegas in a dynamic, in terms of routing, network. In such a case, TCP Vegas is misled by the changing Round-Trip Time (RTT) during a reroute, the increase of which designate congestion for TCP Vegas and thus decreases the congestion window and the utilization of the link. In other words, these misinterpretations lead to instability of the utilization of some links. In order to enhance TCP Vegas functionality and make it more stable, a Network Empowerment Mechanism (NEM[2]) (under development) can offer to TCP Vegas the knowledge if this RTT change was related to congestion or to another reason (e.g. reroute due to a fallen link).

Context 2: Stability issues for a dynamic virtual network

Let's consider an infrastructure encompassing IT resources (e.g. physical servers) and network resources (physical routers). Let's assume that Virtual Machine (VM) and Virtual Router (VR) can be moved from one physical node to another (the physical node merely serves as the carrier substrate on which the actual virtual node operates). Actually VR is a logical router that separates its functionality from the physical platform that hosts the entity, including mechanisms for management, configuration, monitoring and maintenance.

In principle, a dynamic provisioning of virtual resources (VMs and VRs) [17] could allow load and traffic engineering in order to improve performance (e.g. limiting hotspots in the IT resources) and to reduce power consumption in the routers network.

In particular, in case of hotspots in the IT resources, operators can change the allocation or migrate VMs to improve performance (e.g. response time). At the same time, as the network traffic volume decreases, operators can migrate VR to a smaller set of physical routers and shutdown or hibernate unneeded physical routers to save power. When the traffic starts to increase, physical routers can be brought up again and virtual routers can be migrated back accordingly).

In summary, in this example we can see the interactions of two main methods, that have to be coordinated to ensure stability and proper levels of performance: the former is in charge of allocating VM across multiple networks for performance optimization; the latter is in charge of migrating VR a smaller set of physical routers for saving power (by shutting down or hibernating unneeded physical routers). It should be noted that although both methods would be stable if operating alone, without a proper coordination, the combination of the two methods might risk a positive feedback loop.

---

[2] *NEM = a functional grouping of objective(s), context and method(s) where "method" is a general procedure for solving a problem. A NEM is (a priori) implemented as a piece of software that can be deployed in a network to enhance or simplify its control and management (e.g. take over some operations). An intrinsic capability of a NEM is to be deployable and interoperable in a UMF context (in a UMF-compliant network).*

Context 3: Vulnerability Management

From a security perspective, the stability of a network also depends on its capability to prevent vulnerable configurations. In autonomic environments, administrators provide high-level objectives while management operations are delegated to the networks themselves. Along with administration tasks done by humans, changes performed by autonomic entities may generate vulnerable states when following high-level objectives. Even though these changes can operationally improve the environment, vulnerable configurations may be produced increasing the exposure to security threats.

Accordingly, vulnerability management plays a crucial role for maintaining safe configurations on devices in these environments. The assessment of local vulnerabilities on a device requires the investigation of specific states and conditions that may allow an attacker to compromise the system. While black-box techniques, such as network scanning can provide useful security information without requiring specific tools in the device under analysis, grey-box techniques can highly enhance the obtained information by accessing the device itself and inspecting its internal state and particular configurations. For instance, the OVAL language provides a support for describing more than 13000 configuration vulnerabilities (official repository). In that context, a NEM on Vulnerability Management related to this topic is under development in autonomic environments.

In addition to local vulnerabilities, distributed vulnerabilities have also to be assessed over a consolidated view of the network in order to detect vulnerable states that may simultaneously involve two or more devices. Traditional mechanisms perform a global analysis by investigating each network element individually. Even though such approaches can detect sets of vulnerabilities that may allow an attacker to perform a multi-step attack, they do not provide the capability of detecting vulnerabilities that simultaneously involve two or more devices under specific conditions. The underlying problem relies in that each network device can individually present a secure state, but when combined across the network, a global vulnerable state may be produced.

Another example is related to the considerable efforts (as shown by the in the prior art) in developing systems and methods for detecting potential attacks (e.g. based on OSPF vulnerabilities in equipment's OS) before they can affect seriously network stability (e.g. routing stability).

PROBLEM STATEMENT

If future network management will evolve towards a framework (or ecosystem) of pieces of software (implementing methods) interacting each other, then there will be concrete risks of systemic instabilities.

Taking the metaphor of ecosystem as assembly of species, each of these may have feedback mechanisms that could ensure stability if acting alone. However, the overall assembly behaviours may show sharps transitions (due to unintended interactions causing non-linearity) from stability to instability: this risk is more and more probable as the number and strength of interactions among species increase.

Already in today networks there are some potential risks of instabilities, but these risks are still limited and rather well controlled: actually Internet can be viewed as a self-organizing ecosystem (achieved through constrained optimizations actuated through protocols and algorithms), whose level of complexity if still low.

However, as the number of methods being introduced (for easing network management) will increase, also these risks of instabilities will dramatically increase, creating real threats for networks (which will become very complex and dynamic, similar to natural ecosystems). Complexity is an attribute of every system or network which is composed of a sufficient great number of interacting elements: close to certain levels of complexity the system or network becomes brittle and vulnerable and it can switch from one mode to another suddenly, spontaneously (e.g. a phase transition occurs which is detected by an abrupt change of an order parameter when a control parameter is varied across the critical point).

It is well known that self-organization implies potential instabilities: stable configurations can be seen as indicating the presence of sort of attractors. An attractor, in this context, means one of the states of the network where the network settles after starting from a given initial condition. Self-organization needs these attractors to have a sufficient instability to be able to alter in order to adapt to the environment. These potential instabilities (and the state transitions) should be kept under control and directed (to avoid network performance degradations or even worst networks breakdowns).

In summary, there is the need of designing and developing a methodology and practical approach for setting-up, configuring and tearing down sets these multiple methods… etc. in order to achieve network optimization and stability at the same time.

The overall problem of ensuring network stability can be decomposed in a set of sub-problems (this decomposition has also be adopted in the QFD analysis of the use case requirements [20], [21]):

PROBLEMS/NEEDS	**PROBLEMS/NEEDS (lower level of details)**
P1 -To have a run time environment for off-line validation and on-line control of self-* features.	To define and collect a set of models, tools for off-line simulations - emulations and in line control of networks stability and performance Also models of embedded self-* mechanisms that can be used in the UMF for tracing the stability of adaptations of those mechanisms
P2 - To validate self-* features off-line (according to predefined criteria) before network deployment.	To make off-line validations based on simulations - emulations of network stability and performance
P3 - To monitor on-line parameters to assess and predict network stability during network operations.	To collect on-line data, events. Also to have traces (patterns) of safe and stable adaptations, be able to aggregate those traces and be able to use them in network forensics
P4- To analyse-elaborate data about network stability during network operations.	To filter and analyse data to assess and predict behaviour of network in terms of stability and performance. Be able to expand the aggregated history traces
P5 - To decide and actuate network self-stabilization in case of emerging instabilities	To decide the self-stabilization actions for maintaining stability and performance (according to SLA). Support the emergence of collaboration patterns (e.g. by means of triggering collaboration policies (predicates)
	To actuate network self-stabilization decision actions for maintaining stability and performance (according to SLA).
P6 -To decide and de-activate manually self-* features in case of persistent instabilities	To decide conditions for de-activating manually self-* features (e.g. persistent instabilities, network cannot self-stabilized). To define conditions per mechanism (group of mechanisms), in which a Call for Governance must be issued
	P2.6.2 - To predispose for the manual de-activation of self-* features

Table 1. Problem decomposition

Requirements coming out from the problem statement have been regrouped in Deliverables D41 and D42 (common to all project case studies). Deliverables D41 and D42 are public reports available on the project web site: http://www.univerself-project.eu/technical-reports or on request (see Contact section at the end of this document).

MODELLING

This section briefly describes the modelling approach adopted by the case study.

Actor(s)

The actor is the Network Operator. Network Operator should have the possibility to monitor and control network stability. This implies the possibility of enforcing high level policies for self-stabilization or even the possibility of de-activating autonomic methods or control-loops (whose unwanted coupling may have caused instability).

The case study on Networks Stability and Performance mainly concerns Service and Resource Management and Operations.

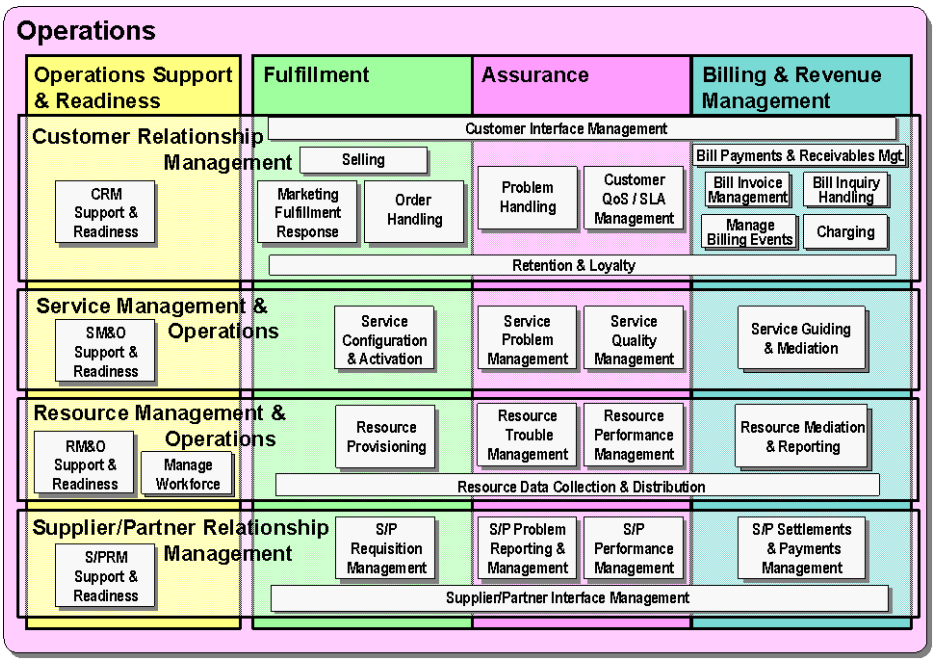

Fig. 2. - eTOM Operations Processes

(picture credits: http://www.tmforum.org/BestPracticesStandards/BusinessProcessFramework/1647/Home.html)

Trigger(s)

The Network Operator should be able to state technical and business goals (e.g. using a human to network interface). Main triggers are:

- Updates of technical and business goals (new deployments of resources or services);
- Critical situations (e.g. hot spots, traffic congestions, performance degradations, etc.);

Phases

Two main phases have been identified:

- Phase A – before network and service deployment
    - Off-line validation
- Phase B – during network operations
    - In-line monitoring and detection of emergence network instabilities
    - Proactive and reactive actions to mitigate emergence network instabilities

INNOVATION

Enabling concepts and mechanisms

The enabling concepts for addressing network stability control concern the mechanisms behind the three complementary approaches currently explored in this use case, which are:

- Preventing vulnerable configurations;
- Design time verifications;
- Network utility functions.

Detection of Vulnerable Configurations

Autonomic networks have to adapt their configurations with respect to their environment, to protect themselves against security attacks, to repair their own failures, and to optimize their various parameters. In order to achieve these objectives, autonomic related operations are performed modifying the environment. Such operations may lead to potential vulnerable states. Preventing vulnerable configurations contribute to the stability of autonomic networks and services. The main activities performed during the lifecycle of the vulnerability management process can be mapped to the same activity line present in autonomic components. First, vulnerability assessment activities take place in the monitoring phase where tasks for analysing and detecting vulnerable states are performed taking advantage of the available security knowledge. When a security problem is found, it is then classified and changes for correcting the situation must be performed. Vulnerability countermeasures may be planned and executed based on several factors such as importance, risks and impact, during the remediation phase.

In that context, we are working on the integration of security knowledge (vulnerability descriptions) into the autonomic management plane. More precisely, our modelling is focused on OVAL vulnerability descriptions using first order logic, and permits the translation of these descriptions into policy rules interpretable by the Cfengine configuration system [7]. The Open Vulnerability and Assessment Language (OVAL) define a specification for vulnerability descriptions and the associated tests [8]. An OVAL description (or OVAL definition) specifies a criterion that logically combines a set of OVAL tests. Each OVAL test in turn defines the process by which a specific configuration condition or property is assessed on the target device. A typical instance of an OVAL test is the checking of the version number of a given library. The overall result for the criterion specified in the OVAL definition is built using the results of each referenced OVAL test. First order logic has been used as an intermediate language to support the mapping of OVAL descriptions into Cfengine policies and to infer a translation algorithm. This work corresponds to the NEM related to vulnerability management. It enables the Cfengine autonomic configuration system to dynamically detect vulnerable configurations when other management operations are executed.

From a more distributed perspective, vulnerability management mechanisms usually perform a global analysis by investigating each network element individually and do not provide the capability of detecting vulnerabilities that simultaneously involve two or more devices under specific conditions. In particular, each network device can individually present a secure state, but when combined across the network, a global vulnerable state may be generated. We are working on an extension of the OVAL language, called DOVAL (Distributed OVAL) [16], by considering a scenario in a VoIP infrastructure where two hosts are involved: a SIP server and a DNS server, each one with specific properties, corresponding to potential exploitable network vulnerability. In this scenario, a denial of service (DoS) attack over the SIP server can be performed by flooding it with irresolvable domain names that must be solved by a local DNS server. The local DNS server in turn, is configured for requesting the resolution of unknown domains to external servers, increasing the number of waiting requests and therefore the response time for each SIP request. Under these configuration states, flooding a SIP server with such type of messages will prevent it to respond to legitimate requests. It is important to highlight that both servers and the relationship between them are required conditions for the distributed vulnerability to be present. If the DNS server is not present or it is not compliant with the required specific conditions, the SIP server would immediately respond to a SIP client that its SIP request has failed.

Design time verifications

Formal verification provides a systematic way to assess the correctness of protocols, processes and systems. The main difference compared to simulation methods is that instead of only examining a limited area of the operational space of the system under consideration, it can be used to examine the whole state space of possible operations and conditions under which the system may operate. This means that all possible

combinations of inputs and actions can be taken into account and, therefore, all possible outputs can be derived and evaluated. One could regard the outcome of formal verification as the outcome of an infinitely large number of simulation runs. This means that contrary to simulations, formal verification methods are capable of capturing conditions and operations that may otherwise remain unnoticed, even after a very large number of simulation runs. In a similar way, formal verification is able to assess the performance and quantify properties of processes with a degree of confidence not possible through simulation runs.

In general there are two approaches in formal verification [10]. The first one is to try to prove the correctness of a system and derive its properties through a sequence of theorems; this is called theorem proving. This process however is very cumbersome in practice and the more complicated the system the harder it becomes to use theorems to prove its correctness. The second approach is called model checking. In this approach the behaviour of the system under consideration is modelled using the description language of the model checker and then the model checker examines all possible system evolutions based on the model. That is model checking itself does not try to "understand" the behaviour of the system but is able to output the outcomes of its behaviour under all possible circumstances and check all of the outcomes with respect to the meeting or violation of properties of interest. The main limitation of model checking is the state explosion problem; as the size of the system and the parameters under consideration increase so do the number of states and transitions between states. To account for this the model of the system should be carefully constructed, omitting operations and features that are not of interest and by "abstracting" appropriately the operations that need to be included in the model. In addition there exist some proposals in the literature on how to scale the applicability of model checking by breaking down large systems into smaller ones, model checking the smaller systems and combining the results to derive the correctness (or faultiness) and properties of the larger system. This approach is known as compositional verification.

In model checking, four main types of models are commonly used, depending on the characteristics of the system to be modelled and analysed; these are Discrete-time Markov Chains (DTMCs), Markov Decision Processes (MDPs), Continuous-time Markov Chains (CTMCs), and Continuous-time Markov Decision Processes (CTMDPs). Decision Processes extend Chains to account for non-deterministic behaviour; that is behaviour where the transition probabilities cannot be clearly defined. For example, probabilities for transitions triggered by external factors at random instances or incurred due to poor/unknown behavioural specification. In the first two models, all transitions take place in discrete (time) steps whereas in the latter two, time is modelled in a continuous manner.

Network Utility Functions

In [11] and [12], Kelly et al. presented an innovative idea of formulating a network optimization problem in terms of maximizing an utility function where the variables are the source rates constrained by link capacities and the objective function captures design goals.

Since then many research activities have been carried out on distributed network resource allocation using the language of Network Utility Maximization (NUM). For example, also cross-layer interactions can be characterized by viewing the process of "layering as decomposition of a given NUM problem into many sub-problems. These sub-problems are "combined together" by certain functions of the primal and dual variables. This framework of "layering as optimization decomposition" is well described in [13].

Utility functions appear to be a valid instrument for addressing the problem of stability and performance of a network. They can be constructed based on user behaviour model, or figures, that can be monitored (and controlled by nodes configurations); example of said figures are: availability, delay, latency, network utilization, network throughput, network bandwidth capacity, network costs, energy consumption, response time, etc.

As an example, TCP/IP protocol can be seen as an example of a protocol for achieving a constrained optimization: its objective is to maximize the sum of source utilities (as functions of rates) with constraints on resources. In fact, each variant of congestion control protocol can be seen as a distributed algorithm maximizing a particular utility function. The exact shape of the utility function can be reverse engineered from the given protocol. Similarly, other recent results also show how to reverse engineer Border Gateway Protocols (BGPs) as a solution to the Stable Path Problem, and contention-based Medium Access Control (MAC) protocols as a game-theoretic selfish utility maximization [14].

In line with the NUM approach, network problems can be decomposed and methods (e.g. distributed algorithms and/or protocols) can be developed, by controlling local variables.

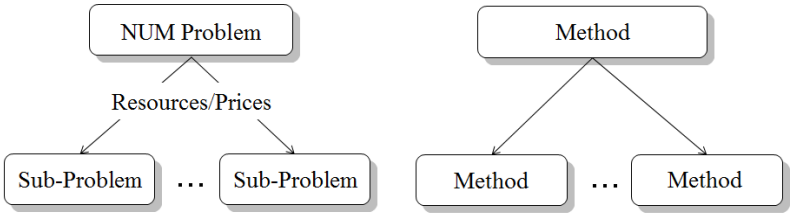

Fig. 3. Problem decomposition vs. methods as CO solvers

It should be noted that by techniques such as Lyapunov function or the descent lemma, global or local asymptotic convergence towards the optimum could be proved. This makes NUM modelling approach very interesting for looking optimization of performance and stability.

Following this modelling, the case study is developing a NEM in charge of coordinating diverse methods in order to solve CO problems whilst avoiding network instabilities.

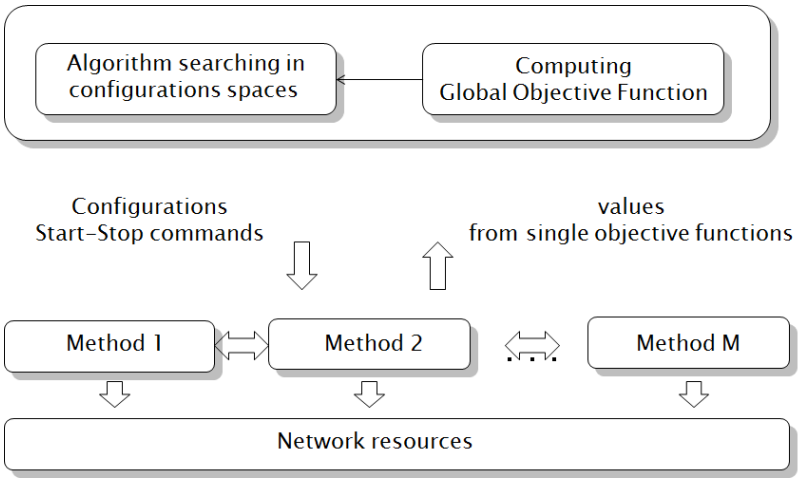

Fig. 4. NEM - Network Stability and Control

Each method tries maximising its associated objective function. At the same time, NEM Network Stability and Control searches, in the space of methods configurations (using, for example, a beam-search algorithm), those configurations allowing maximising the global objective function (which is a combination of the single utility functions). This is done at regular intervals, upon reaching a trigger or in reaction to changes in the global objective function.

Differentiation from the state of the art

Today, the issue of network stability is only partly covered. In the future, higher and higher complexity and dynamicity of networks will pose serious problem of understanding, monitoring and controlling network instabilities.

A methodology (definitions, models and methods) and the related instruments (tools) are still missing to allow Network Operators to validate (off-line) to monitor (in-line), predict and control the emergence of network instabilities. Some solutions have been developed for specific areas of applications, e.g. dynamic routing, queue control, flow control. Nevertheless a systematic approach to network stability is still missing.

The case study solutions will lay the foundations of this systematic approach.

Progresses with respect to the state of the art include:

- Definition of a network stability metrics (how measure stability and how to manage it);
- Definition of a general methodology and practical approaches for controlling network instabilities due to the potential unwanted coupling or even competition of multiple methods;

- Development open source tool-boxes for simulations-emulations of stability and performance of networks with (large) number of nodes.

Impacts and benefits

Future networks management will require the introduction of “self-organizing” capabilities. Practically this will be achieved through protocols, control loops, algorithms… etc. exploited across network layers.

This will impact traditional network and service management by reducing CAPEX and OPEX: load balancing and traffic engineering, for example, not only can improve network performances (e.g. IT response time or net throughput), but can also ease human operators in Operations (reduction OPEX) and postpone network resources investments (reduction CAPEX).

In order to maximize the benefits from introducing these methods, or better NEMs, it is necessary to orchestrate their actuations in order to avoid potential unwanted coupling or even competition causing instabilities. The case study solutions impact and benefit will be measured in terms of optimizing said benefits, by coordinating the methods in order to avoid (or limit) competitions and instabilities. Quantifications of the benefits could be done case by case through the adopted utility or objective functions.

TO BE CONTINUED

This document is the first part in a series covering the introduction, general description, problem statement and innovation of the case study Networks Stability and Performance. Subsequent and complementary parts will be published in the near future, during the lifetime of the project with even more information, results and innovations.

Keep in touch to get premium access to these future reports!

REFERENCES

[1] T. Ohira and R. Sawatari, Phys. Rev. E 58, 193 (1998);

[2] R. V. Sole, "Information Transfer and Phase Transitions in a Model of Internet Traffic", Physica A (2001), Volume: 289, Issue: 3—4: Publisher: Elsevier BV, Pages: 595-605;

[3] V. Marbukh, "Towards Understanding of Complex Communication Networks: Performance, Phase Transitions & Control", Sigmetrics Performance Evaluation Review, 2007;

[4] B. H. Wang, "Routing strategies in traffic network and phase transition in network traffic flow", Pramana, Journal of Physics, Vol. 71, n.2, August 2008;

[5] B. Ford, "Icebergs in the Clouds: the Other Risks of Cloud Computing", available at arXiv:1203.1979v1;

[6] K. Korovin, "Logic in computer Science", lecture notes available at http://www.cs.man.ac.uk/~korovink/cs2142/;

[7] , R. Badonnel, and O. Festor. Supporting Vulnerability Awareness in Autonomic Networks and Systems with OVAL. Proceedings of the ACM SIGCOMM / IFIP / IEEE International Conference on Network and Service Management (CNSM'11), October 2011.

[8] OVAL, Open Vulnerability and Assessment Language, Mitre Corp, http://oval.mitre.org/;

[9] M , R. Badonnel, and O. Festor. Towards Vulnerability Prevention in Autonomic Networks and Systems. Proceedings of the IFIP International Conference on Autonomous Infrastructure, Management, and Security (AIMS'11), PhD Workshop, June 2011 ;

[10] Cfengine, Cfengine AS, http://www.cfengine.org/;

[11] F.P. Kelly, " Charging and rate control for elastic traffic". European Transactions on Telecommunications 1997; 8:33–37;

[12] F.P. Kelly, A. Maulloo, D. Tan, "Rate control in communication networks: shadow prices, proportional fairness and stability". Journal of the Operational Research Society 1998; 49:237–252;

[13] M. Chiang, S. H. Low, A. R. Calderbank, J. C. Doyle, "Layering As Optimization Decomposition: A Mathematical Theory of Network Architectures", Proceedings of the IEEE, Vol. 95, No. 1. (05 January 2007), pp. 255-312, doi:10.1109/JPROC.2006.887322;

[14] M. Chiang, "Balancing transport and physical layers in wireless multihop networks: Jointly optimal congestion control and power control," IEEE J. Sel. Areas Commun., vol. 23, no. 1, pp. 104–116, Jan. 2005;

[15] A. Manzalini "Mitigating Systemic Risks in Future Networks" accepted by IEEE 17th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks 2012 at CAMAD 2012;

[16] M. Barrere, R. Badonnel, and O. Festor,. Towards the Assessment of Distributed Vulnerabilities in Autonomic Networks and Systems, Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS'2012), USA.

[17] Clayman, S., Clegg, R., Galis, A., Manzalini, A. "Stability in Dynamic Networks"– Future Network and Mobile Summit 2012, Berlin, 4-6 July 2012, http://www.futurenetworksummit.eu/2012/

[18] Jacobson Ivar, Christerson M., Jonsson P., Övergaard G., Object-Oriented Software Engineering - A Use Case Driven Approach, Addison-Wesley, 1992

[19] Cockburn, Alistair. Writing Effective Use Cases. Addison-Wesley, 2001.

[20] "Synthesis of use case requirements" - Deliverable 4.1, August 2011, http://www.univerself-project.eu/technical-reports

[21] "Synthesis of use case requirements – Release 2" - Deliverable 4.2, April 2012, http://www.univerself-project.eu/technical-reports

CONTACT INFORMATION

For additional information, please contact: Antonio Manzalini (antonio.manzalini@telecomitalia.it); or consult www.univerself-project.eu

UNIVERSELF CONSORTIUM