

OSPF for Implementing Self-adaptive Routing in Autonomic Networks: a Case Study

Gábor Rétvari¹, Felicián Németh¹, Ranganai Chaparadza², Róbert Szabó¹

¹ Department of Telecommunications and Media Informatics
Budapest University of Technology and Economics, Budapest, Hungary
{retvari,nemethf,robert.szabo}@tmit.bme.hu

² Fraunhofer Institute for Open Communication Systems, Berlin, Germany
ranganai.chaparadza@fokus.fraunhofer.de

Abstract. Autonomicity, realized through control-loop structures operating within network devices and the network as a whole, is an enabler for advanced and enriched self-manageability of network devices and networks. In this paper, we argue that the degree of self-management and self-adaptation embedded by design into existing protocols needs to be well understood before one can enhance or integrate such protocols into self-managing network architectures that exhibit more advanced autonomic behaviors. We justify this claim through an illustrative case study: we show that the well-known and extensively used intra-domain IP routing protocol, OSPF, is itself a quite capable self-managing entity, complete with all the basic components of an autonomic networking element like embedded control-loops, decision-making modules, distributed knowledge repositories, etc. We describe these components in detail, concentrating on the numerous control-loops inherent to OSPF, and discuss how some of the control-loops can be enriched with external decision making logics to implement a truly self-adapting routing functionality.

Keywords: Autonomic/Self-managing Networks, self-adaptation, autonomic routing functionality, OSPF

1 Introduction

OSPF (Open Shortest Path First, [1]) is perhaps the most successful routing protocol for the IP suite. It is a link-state routing protocol, with two-level routing hierarchy, support for basically any type of medium IP supports, an integrated neighbor discovery and keep alive protocol, reliable link state flooding, fast shortest path routing algorithm, support for multipath routing, etc. Accordingly, OSPF rapidly gained ground after its inception, and it has been enjoying unparalleled popularity in the networking community to our days. Only special environments, like extremely large ISP backbones or heterogeneous, multiprotocol networks, are where alternatives are preferred [2].

A key enabler in the success of the OSPF routing protocol is the inherent capability for self-management built into it from the bottom up. Self-management,

in the case of OSPF, means that the routers participating in the distributed process of IP routing perform various management tasks autonomously, independently of any higher level control function or manual intervention. A quintessential self-management operation built into OSPF is self-adaptation to the network topology at hand: OSPF routers autonomously discover network topology, disseminate topology information and compute shortest paths to produce consistent routing tables, and this self-adaptation mechanism is completely independent of external control or human supervision of any kind. There are several other self-management functionality hard-wired into OSPF, like autonomous detection of neighbors (auto-discovery), autonomous advertisement of capabilities (self-advertisement), adaptation to failures and outages (self-healing), etc. Thanks to these sophisticated self-* features, only little manual configuration is required for OSPF to perform the complex task of routing table maintenance. Once network interfaces are properly brought up and supplied with unique IP addresses, and OSPF is made aware of the interfaces it should use, the protocol is up and running, with full support for basic network routing. Only for complex operations, like link weight adjustment, multiple administrative areas, or interfacing with inter-domain routing protocols, does OSPF need special configuration and manual intervention on the part of the network operator.

One could argue, however, that in fact the autonomous capabilities built into OSPF are a bit over the point, as sometimes these self-management functions may work against, instead of cooperating with, the network operator. This is because for a network operator to achieve her network-level performance objectives, she needs to be in full control of the network. However, once OSPF with its intrinsic self-management functionality comes into the picture, the management actions taken on the part of the network operator and the self-management actions carried out by OSPF might easily end up interfering with each other. For instance, the policy of OSPF to provision the forwarding paths exclusively over shortest paths might differ from the policy seen by the operator as optimal. Thus, network operators have for a long time been working around OSPF's shortest path routing by tweaking the link weights for obtaining the desired routing pattern [3]. In other words, fulfilling network-level objectives is much easier with direct control over the configuration of the data plane and therefore, the decision logic should not be hard-wired into protocols but rather it should be re-factored into an external, pluggable control logic, which should be fed by monitoring information from the network and whose output should then be communicated back to the routers. This modularization of network fabric and control intelligence is one of the most important promises of the concept of Autonomic Networking.

For anyone to be able to incorporate OSPF into an autonomic network framework, one must need to be aware of the inherent self-management functionality built into it. Without first discovering how OSPF performs autonomous adaptation to external and internal stimuli by itself, anyone willing to deploy autonomic networking functionality on top of OSPF might easily find himself needlessly re-implementing autonomic functionality already present in OSPF, or blindly inter-

fering with OSPF's intrinsic control-loops. The aim of this survey is, therefore, to reinterpret OSPF in an autonomic networking context. In other words, we are curious as to how well OSPF fits into the autonomic networking framework.

In what follows, we shall use the GANA framework (Generic Autonomic Network Architecture [4]), proposed by the EFIPSANS project [5] recently, as reference model for autonomic networking. GANA is an architectural Reference Model for Autonomicity and Self-Management within node and network architectures. In [4], different instantiations of the GANA approach for different types of network devices and network environments (e.g., wireless, mobile or wired/fixed) are presented, which demonstrate its use for the management of a wide range of functions and services, including both, basic network services such as autonomic Routing and autonomic Monitoring, as well as advanced services such as autonomic Mobility and Quality of Service (QoS) Management. A central concept of GANA is that of an autonomic Decision Element (DE) that implements the logic that drives a control-loop over the management interfaces of its associated Managed Entities (MEs). Thus, in GANA, self-* functionalities are always associated with certain control-loops, implemented by Decision Element(s). Additionally, GANA organizes DEs into a DE-hierarchy: the higher we go up the hierarchy of DEs, the broader the global knowledge that is required by a DE to take decisions on managing/controlling its associated MEs, which may in turn inductively trigger actions on lower level MEs down to the level of protocols.

The rest of the paper is structured as follows. First, we describe the self-* functionality embedded in OSPF and we treat the most important intrinsic control-loop of OSPF, the self-adaptation control-loop, in large detail (Section 2). We also discover the state of the art in various extensions and improvements to this intrinsic self-adaptation control-loop. The second part of the paper (Section 3) is devoted to demonstrate how OSPF could be exploited for implementing self-adaptive routing in the GANA framework, what Decision Elements and control-loops would be necessary, and we delve into some implementation details. Finally, we conclude the paper in Section 4.

2 Self-* functionality in OSPF

Routing is the process of ensuring global reachability between IP routers and hosts. In the beginning, routing tables were provisioned manually. This practice led to daunting management complexity as networks grew, and it was quite prone to human errors. The very purpose of introducing routing protocols was to mitigate this management burden by automating the process of setting up routing tables. In some sense, autonomic networking can be seen as an extension of this idea to the extreme: make the systems themselves tackle all the management complexities, not just routing, that are otherwise difficult to handle manually. For this, the network needs to be able to self-provision and self-manage to some extent. Such self-* functionality involve, for instance, self-adaptation to changes in the operational conditions, self-healing to repair or circumvent failures, self-optimization for improving performance related aspects of networking, etc. Be-

low, we show that many of these self-* functionalities, either only partially or in their entirety, can readily be identified in one of the most commonly known and used routing protocols, the Open Shortest Path First (OSPF) routing protocol [1]. The enumeration is given in roughly the same order as the corresponding self-* function appears in the course of the real operation of the protocol.

Auto-discovery: OSPF routers discover their immediate neighborhood by using the Hello protocol. Neighboring routers periodically exchange Hello packets, so that each router is aware of all the routers connected to any one of the links or LANs attached to its interfaces. This auto-discovery mechanism is pretty extensive, covering every aspect of routing except for one very important issue. In particular, it lacks support for box-level discovery. This means that routers do not autonomously self-detect their interfaces, and manual configuration is needed to make an OSPF router aware of the interfaces it should involve in OSPF message passing (see more on this issue later).

Self-description: OSPF routers self-describe their capabilities in the Hello packets they generate. Hello packets contain information regarding the highest version of OSPF the router's implementation supports, plus a bitfield to describe OSPF extensions the router understands. This makes it possible to eliminate the chance of mis-configuration arising from letting routers using divergent OSPF versions to speak to each other, or to deploy protocol extensions seamlessly.

Self-advertisement: routers generate so called Link State Advertisements (LSAs) to advertise their forwarding services into the network. These LSAs convey information on the individual routers making up the topology, with their identity, attached interfaces, IP addresses, etc.; the links and LANs connecting the routers with their type (broadcast, Non-Broadcast-Multiple-Access, point-to-point); administrative link cost, etc. OSPF variants, like Traffic Engineering extensions to OSPF (OSPF-TE, [6]) and OSPF extensions in support of Generalized Multi-Protocol Label Switching (OSPF-TE-GMPLS, [7]) add their respective type of link state information to LSAs, involving the link's transmission capacity, free capacity, protection type, the forwarding services offered by the router, the multiplexing/demultiplexing capability of interfaces, etc. It is by passing these LSAs around between routers using a reliable, acknowledged flooding protocol that OSPF synchronizes routing information across the domain. This mechanism basically maintains a distributed, versioned, massively parallel Link State Database (LSDB), shared and synchronized amongst OSPF routers, which ensures that (in steady state) each router holds exactly the same copy of the network topology and thusly consistent forwarding paths are selected.

Self-configuration and self-organization: self-configuration involves setting up and maintaining some configuration parameter by the protocol itself, that otherwise would be handled by the network operator. Self-organization is a method by which entities autonomously organize themselves into groups or hierarchies. A good example of self-configuration and self-organization in OSPF is the election of Designated Routers. In order to reduce the amount of protocol traffic on LANs that connect multiple routers, each OSPF router synchronizes with only a single neighbor, the Designated Router (DR), instead of having to exchange

signaling information with all the other neighbors. The DR is responsible for generating an LSA on behalf of the LAN. The actual DR (and the Backup DR, which is just what its name says) is elected autonomously, by means of the Hello protocol, without the need to be configured manually by the network operator.

Self-healing: an operational network is constantly subjected to disturbances from the environment, most important amongst these is the intermittent and unavoidable failures of network devices and transmission media. To come over failures, OSPF implements a simple but efficient self-healing mechanism: once a router detects that a certain node or link went down (through not getting Hello packets for a certain amount of time from a specific direction), it advertises the changed topology information into the domain, leading to a global recalculation of routing tables with the failed node or link removed from the topology. Although this self-healing mechanism might be somewhat slow due to the need to deliver the LSA to the furthest part of the network to achieve correct global response, it is highly effective in maintaining reachability as long as the network remains connected, irrespective of the number and type of failures occurring.

Self-adaptation: the most important self-management functionality implemented by OSPF is undoubtedly self-adaptation to the topology at hand. This means that routes are not provisioned statically, but instead OSPF is able to dynamically maintain correct, consistent and loop-free forwarding tables over an arbitrary topology. Self-adaptation is, therefore, the most important property of OSPF and the very purpose the protocol was designed in the first place. In the next section, we shall discuss the self-adaptation control-loop in more detail.

One could as well keep on listing the various self-* functionality built into OSPF further (e.g., self-protection, etc.), but we believe that the above examples were sufficient enough to demonstrate the richness of OSPF in terms of autonomy. In fact, OSPF implements, either in its entirety or only partially, pretty much every possible self-* functionality, with the notable exception of box-level auto-discovery and self-optimization. The lack of self-optimization means, for instance, that OSPF is not able to readjust forwarding paths (or at least, the link costs) in order to mitigate or eliminate congestions or load-balance traffic, that is, to optimize the performance of the network. In order to equip OSPF with self-optimization functionality, we shall need to incorporate it into an external control-loop, as shall be discussed later on.

2.1 Self-adaptation in OSPF

Self-adaptation to the underlying topology, to topology changes and to other stimuli effecting network routing, is the main purpose of a routing protocol. Below, we interpret the main self-adaptation control-loop implemented by OSPF in the context and terminology of the GANA framework for autonomic networks [4].

The basic operation of the self-adaptation control-loop of OSPF is as follows. When OSPF takes off or when the underlying topology changes (that is, when a link or node goes down or comes up again), the auto-discovery mechanism of OSPF (i.e., the Hello protocol), through observing the neighborhood of the

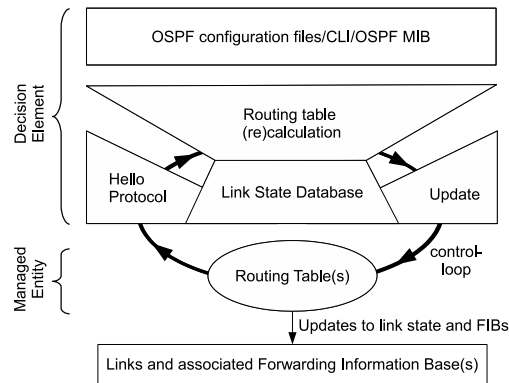


Fig. 1. Self-adaptation control loop in OSPF.

router, initiates the self-advertisement functionality to flood the (changed) network state information throughout the network. Routers, upon the receipt of new LSAs, calculate new routing tables based on refreshed routing information, which makes it possible to always self-adapt to the actual network topology.

In GANA, this self-adaptation mechanism is a typical example of a *protocol-intrinsic control-loop*, with the OSPF protocol acting as a virtual distributed Decision Element (DE) scattered all over domain. The following important autonomic networking components are involved in this control-loop (see Fig. 1):

- The *Managed Entity* is the collective set of all Routing Tables in the network, which are then used to populate the Forwarding Information Bases (FIBs). The Routing Table(s) is the entity on which the control-loop’s effector operates.
- *Monitoring* of the links/interfaces and their forwarding state is implemented by the Hello protocol
- *Analyze/Plan* is broadly mapped to the process of routing table (re)calculation
- *Execution* corresponds to the process of updating the Routing Tables as well as the FIBs across the routers with the next-hops obtained during the last routing table calculation
- *Knowledge* is manifested in this self-adaptation control-loop by the collective set of the LSDBs of the routers. Note that, however, OSPF pays special attention to always keep the LSDBs at each router consistent, synchronized and up-to-date, therefore the Knowledge component is in fact replicated throughout the network instead of being unified at a central knowledge base.
- *Sensors/Effectors for higher level DEs*: the OSPF protocol supplies a number of interfaces for higher level DEs to interact with it, including (usually) a set of OSPF configuration files, a Command Line Interface (CLI) plus a versatile, standardized Management Information Base (MIB).

2.2 Extensions to the intrinsic Self-adaptation control-loop of OSPF

The self-adaptation mechanism built into OSPF is somewhat basic, although quite straight-to-the-point: it only involves tracking topology changes. However, it does not contain mechanisms, knowledge, services, algorithms or other protocol machinery to self-adapt other aspects of routing to the actual environment, or to adapt the self-adaptation control-loop itself when changing conditions make it necessary. Below, we give a brief overview of the state-of-the-art as to how the research community proposes to extend the basic self-adaptation control-loop of OSPF beyond its present capabilities. Note that the survey below lists only those proposals that build the extensions right into OSPF itself, and do not involve those ones that organize OSPF into an external control-loop with a separate Decision Element (we discuss the latter option in the next section)

Self-adaptation to changing resource availability: it has been pointed out many times as one of the major shortcomings of OSPF-type link-state routing protocols that their self-adaptation mechanism is static, as it only involves adaptation to the (changing) topology, but not to the changing operational conditions, the amount and type of actual ingress or egress traffic, availability of free resources, QoS criteria, Service Level Agreements, customer-provider and peering relationships, etc. This makes OSPF routing insensitive to the fluctuations of user traffic, or its changing/evolving embedment into a vibrant socio-cultural/economic environment. The result is often suboptimal routing: traffic tends to concentrate along shortest paths while, at the same time, complete portions of the network remain under-utilized. Various attempts have been made to introduce some forms of dynamic routing into OSPF, by adding further information to the link state, most importantly, the amount of provisionable resources at network elements, and selecting routes that avoid overloaded components. A good example is the Quality of Service extensions to OSPF (QoSPF, [8]), or recent standardization efforts, like OSPF-TE and OSPF-TE-GMPLS.

Self-adaptation and multipath routing: the basic mode of operation in OSPF is to select the shortest weighted path towards a destination, where path weight is understood in terms of some administrative cost set for each network link. Where ambiguity arises (that is, when there are more than one shortest paths to a destination), one path is selected somewhat randomly. In the Equal-Cost-MultiPath (ECMP) mode, however, a router is allowed to use all the potential shortest paths to the destination, by splitting traffic roughly equally amongst the available next-hops. Unfortunately, however, OSPF-ECMP still does not qualify as a self-adaptive multipath routing protocol, because traffic is not balanced with respect to available resources along the paths. Additionally, the path-diversity of ECMP is somewhat insufficient, as in many topologies it is only a rare accident that multiple shortest paths become available to a destination. OSPF-Optimized-MultiPath (OSPF-OMP, [9]) is aimed at overcoming these difficulties: it improves path diversity by not confining itself to shortest paths but utilizing all loop-free paths instead, and it makes self-adaptation sensitive to varying operational conditions by dynamically readjusting traffic shares at indi-

vidual forwarding paths with respect to actual resource availability along those paths.

Fast self-adaptation and self-healing: as mentioned earlier, the self-adaptation control-loop in OSPF involves a tedious global resynchronization of Link State Databases plus additional recalculations of the routing tables. This scheme of global, reactive response makes the convergence of the control-loop slow, which yields that the reaction to failures is a lengthy process in OSPF. Furthermore, the auto-discovery process, responsible for detecting the error in the first place, adds its own fair share of sluggishness to the process (the smallest granularity of the Hello timers is 1 sec, basically increasing the convergence time to the order of seconds). To speed up convergence, a localized, proactive approach should be taken instead, and this is exactly the way the IP Fast ReRoute (IPFRR, [10]) suite of standards is set to remedy the situation. In IPFRR, OSPF pre-computes detours with respect to each potentially failing component and stores the next-hops in an alternative forwarding table. By using an explicit, fast detection mechanism like Bidirectional Forwarding Detection (BFD, [11]), discovering a failure is possible within milliseconds of its occurrence. If an error is detected, OSPF switches to this alternative table and suppresses global response by withholding the fresh LSA in the hope that the failure is transient. Should the failure go away soon after, OSPF switches back to the original forwarding table as if no failure happened. Only when a failure persists for a longer period of time, global re-convergence is initiated. IPFRR proved highly efficient in practice, bringing down to failure recovery to the order of milliseconds [12].

Self-adaptation with partial/outdated link-state information: a basic assumption lying in the heart of the design of OSPF is that the information in the Link State Database is always consistent and up-to-date. Otherwise, self-adaptation might suffer as certain destinations might become unreachable and transient or even persistent routing loops might emerge. The assumption of consistency and freshness, however, might not always hold: in fixed or slowly changing wireless networks, it would be crucial to limit the amount of signaling traffic exchanged between nodes in order to save battery power. The idea here is to tweak the self-advertisement mechanism so that routers hold precise information only in a limited vicinity, and the further they look the more inaccurate the link state information. As a packet travels hop by hop in the network, it will always see locally accurate link state, leading to, hopefully, close to optimal shortest path forwarding. For pointers on how to change the intrinsic self-adaptation of OSPF, see FishEye State Routing [13] or XL Link State [14].

3 OSPF for implementing truly self-adaptive routing

So far, we have seen that OSPF implements a solid number of self-* functions in itself, by means of control-loops embedded deep into the protocol machinery. The most important of these, the self-adaptation control-loop, is an efficient and robust control-loop and, considering the numerous extensions to this control-loop on their way to standardization and large-scale deployment, OSPF

is expected to need only very little governance from higher layers to achieve its full potential. *This does not mean that OSPF on its own qualifies as an autonomous, self-managing protocol entity, only that certain functionality it embeds can be interpreted, to some extent, within the context of autonomous networking frameworks, like GANA.* Instead, to truly realize the vision of autonomous, self-managing routing, separate Decision Element(s) are needed, decoupled from OSPF and implemented in higher layers of the GANA DE-hierarchy, either because, due to implementation considerations, decision making would be very difficult to engineer into OSPF itself, or the decision making process would make use of external information that is simply not available at the lowest level of the DE hierarchy OSPF resides at in the GANA architectural Reference Model. Or, one might deliberately choose not to embed certain control-loops into OSPF in order to better modularize the architecture, to separate managed and managing functionality from each other, to easily swap/change the decision making process, etc.

Next, we discuss how Autonomous Routing is modelled in the GANA framework, and then we describe some additional control-loops to realize this vision on top of OSPF.

3.1 Implementing Autonomous Routing following the GANA approach

The Routing Functionality of network nodes and the network as whole can be made autonomous by making diverse Routing Schemes and Routing Protocol Parameters employed and altered based on network-objectives, changes to the network's context and the dynamic network views in terms of events, topology changes, etc. Fig. 2 depicts how the routing behavior of a node/device and the network as a whole can be made autonomous. The cloud on Fig. 2 represents an overlay or logically centralized DE(s): (1) With wider network-wide view to perform sophisticated decisions, e.g., network optimization; (2) Centralized to either avoid processing overhead in managed nodes or scalability and/or complexity problems with distributed decision logic in network elements; (3) The Elements in this cloud may be the ones that provide an interface for a humans to define network Goals and Objectives or Policies, e.g., Business Goals.

Two types of control-loops are required for managing/controlling the routing behavior. The first type is a node-local control-loop that consists of a *Function-Level Routing_Management_DE* embedded inside an autonomous node, e.g., a router. The local *Function-Level Routing_Management_DE* is meant to process only that kind of information that is required to enable the node to react autonomously and autonomously (according to some goals) by adjusting or changing the behavior of the individual routing protocols and mechanisms required to be running on the node. It reacts to "views", such as "events" or "incidents", exposed by its Managed Entities (MEs), i.e., the underlying routing protocols or mechanisms. Therefore, the *Routing_Management_DE* implements self-configuration and dynamic reconfiguration features specific to the routing functionality of the autonomous node.

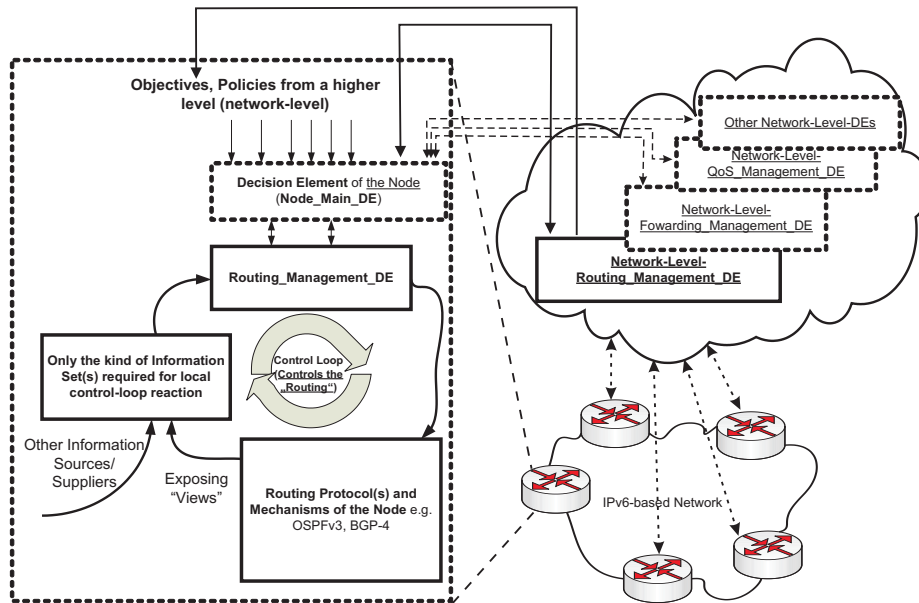


Fig. 2. Autonomicity as a feature in Routing Functionality.

It is important to note that due to scalability, overhead and complexity problems that arise with attempting to make the *Routing_Management_DE* of a node process huge amounts of information/data for the control-loop, logically centralized DE(s), outside the routing nodes, may be required, in order to relieve the burden. In such a case, a network-wide slower control-loop is deployed in addition to the faster node-local control-loop (with both types of loops working together in controlling/managing the routing behavior). Therefore, both types of control-loops need to work together in parallel via the interaction of their associated *Routing_Management_DEs* (one in the node and another in the realm of the logically centralized network overlay DEs). The *Node-scoped Routing_Management_DE* focuses on addressing those limited routing control/management issues for which the node needs to react fast (the faster control-loop). At the same time, it listens to control from the *Network-level Routing_Management_DE* of the outer, slower control loop, which has wider network-views and dedicated computational power, and thus is able to compute routing specific policies and new parameter values to be used by individual routing protocols of the node. The *Network-level Routing_Management_DE* disseminates the computed values/parameters to multiple node-scoped *Function-Level Routing_Management_DEs* of the network-domain that then directly influence the behaviour of the targeted MEs, that is, the routing protocols and mechanisms within a routing node. The interaction between the two types of *Routing_Management_DEs* is achieved through the *Node_Main_DE* of a node, which verifies those interactions against the overall security policies of the node. The

Node-scoped Routing-Management-DE also relays “views”, such as “events” or “incidents”, to the *Network-level Routing-Management-DE* for further reasoning.

3.2 Control-loops for Autonomic routing on top OSPF

Auto-discovery: as mentioned earlier, auto-discovery in OSPF lacks box-level discovery. This means that there is no way for an OSPF router to learn autonomously the set of functional interfaces it has, and configure the protocol properly on those interfaces. Unless OSPF is explicitly told so (by means of manual configuration, intervention at the CLI or through central network management), an OSPF router will not use an interface, even if, by all measures, it legitimately could. The reasons for this are multi-faceted. First, whether to run or not run OSPF on a specific interface is a crucial configuration issue, dependent on high-level network engineering policies, and it has important security implications as well. There are various configuration parameters (e.g., timers, link cost, authentication, link type, etc.) that quite commonly need manual tweaking too. Additionally, every interface must uniquely be associated with a specific OSPF area, retaining the consistency of the network via the Backbone Area, and this also must be handled through explicit configuration. Therefore, OSPF must be involved in some forms of a auto-discovery/self-configuration control-loop: an OSPF router needs to send its identity and some router-specific data upstream to the *Network-level Routing-Management-DE*, and in turn it gets back a routing profile with a complete set of configuration parameters to set, including the list of interfaces to bootstrap, timer settings, authentication keys, link costs, etc.

Self-configuration: as discussed previously, the network operator can tweak the operation of OSPF, with considerable detail and granularity, through configuration parameters. Monitoring, setting, and adjusting from time to time these configuration parameters might benefit the operation and the performance of the network. The main enabler for this is OSPF’s timer mechanism, which allows the network operator to manipulate the way certain events are scheduled.

For instance, in networks with limited resources the signaling traffic generated by OSPF might put too much burden on the network infrastructure, leading to premature drainage of battery power, congestion along low-capacity links, etc. Thus, relying on monitoring data, a higher level DE might instruct OSPF to slow down LSA propagation (OSPF parameter: `MinLSInterval`, `MinLSArrival`³ and `RxmtInterval`, see Appendix A and C in [1]). The Hello protocol could also be tweaked to generate smaller keep-alive traffic in slowly changing environments or between high-reliability interfaces where failures only rarely show up (OSPF parameter: `HelloInterval`, `RouterDeadInterval`). To save precious battery power at wireless nodes, OSPF routing table recalculations could also be throttled leading to smaller computational burden on the CPU (note that

³ These two parameters are protocol constants, or architectural constants, which means that they are not marked as user configurable in the OSPF standard, though, they could be easily made to be so in implementations.

the OSPF standard does not provide an OSPF parameter to throttle SPF calculations, but implementations usually do).

Self-optimization: in OSPF, link weights completely determine, through the shortest path algorithm, the emergent forwarding paths and thus have crucial impact on the way user traffic flows through the network. Even though link costs are set to a constant value almost universally in today's OSPF deployments, it does not necessarily have to be so. Constant weight setting (either to a default constant or a static value reciprocal to the link capacity) leads to a routing that is completely and adversely independent of the operating conditions (i.e., the load at network links, the amount of incoming user traffic, etc.), which often leads to inefficient data forwarding.

In order to improve the performance of the network, link costs could (and should) be regularly updated to better reflect the actual operational circumstances, and this could be done by organizing OSPF into a self-optimization control-loop. In this control-loop, either the router itself (in particular, the router's *Node-scoped Routing_Management_DE*) or the *Network-level Routing_Management_DE* could readjust the link costs (OSPF parameter: **Interface output cost**). The difference between the two approaches is the amount of monitoring data based on which link cost recalculation can be done. If the link costs are readjusted at the node level, only knowledge available to that node can be used in the readjustment. For instance, cost of overloaded or heavily used local interfaces could be increased, while the cost of underutilized local links could be decreased, but no information on the load at links at remote parts of the network could be used in this process. Unfortunately, careless local intervention, apart from flooding the network with excess signaling traffic, usually leads to global oscillations and instability [15]. Hence, it is better to readjust link weights at the network level [3], [16]. It is important to note that, as theoretical results confirm, it is hopeless to drive the network right to optimal performance, due to the inherent limitations and inflexibility of shortest path routing. Yet, one can go pretty close to the optimum [3], and this is more than enough in most of the cases.

Self-organization: routers in an OSPF domain can be structured into so called areas, with complete topology information available inside the area but only limited, summarized information flooded between the areas. This makes it possible to reduce the signaling overhead of OSPF, better modularize the network, eliminate excess signaling in stub areas (parts of the network with a single ingress/egress router), etc. In a bandwidth-constrained network, the area structure could be readjusted every time it is believed that the signaling overhead of OSPF could be reduced that way. Note that, however, changing area boundaries on the fly may lead to routing-loops and transient spillage of reachability, making this self-organization function less appealing. Curiously, even the *raison d'être* of areas has been questioned quite vigorously in the near past [17].

Inter-domain self-adaptation: interaction between intra-domain and inter-domain routing has for a long time been a hardly understood and thus heavily researched question [18]. The two are not completely independent from each other:

a simple change of an intra-domain forwarding path might have far-reaching consequences, causing routing updates in a significant portion of the entire Internet. Additionally, many of the changes of external routes are advertised into the routing domain, often causing the fluctuation of ingress traffic and/or outgoing traffic. This makes it important to try to readjust the interaction between intra- and inter-domain routing based on a detailed knowledge of topology, monitoring data, network mining data, predicted profiles, AS paths, BGP policies, etc. However, currently there exists very little understanding as to how changing some aspects of this interaction affects the rest of the network, which makes designing, implementing and optimizing such an inter-domain self-adaptation control-loop quite a task.

4 Conclusions

In this paper, we have seen that one of the most widely known network protocols, the OSPF routing protocol, includes a fair number of traits that qualify as some forms of self-management. Apart from the main control-loop, the self-adaptation control-loop, we have shown examples of auto-discovery, self-healing, self-configuration, etc. Taking a wider look, it turns out that many network protocols in existence today, especially telecommunication protocols, are designed around some forms of a control-loop. A good example is the venerable TCP flow control protocol. However, these control-loops were designed piece-meal and were deeply hidden in the very substrate of the network protocol engines. A basic promise of the emerging concept of Autonomic Networking is to give a new and enlightening way to think about network-management, thus facilitating for implementing advanced and enriched self-manageability of network devices and networks. This allows us to re-engineer *implicit* control-loops, that are currently buried deeply into many protocol engines, into *explicit* control-loops, communicating over standardized interfaces with heavily optimized and flexible decision making logics changeable and swappable on the fly.

In the case of OSPF, we undertook the first steps of this process. We identified the most important control-loops intrinsic to OSPF, which makes it possible to open up the protocol and incorporate it into external control-loops, where necessary. We found that very little external control is needed for the basic operation of OSPF. One such issue is auto-configuration with proper routing profiles for bootstrapping. In the case of self-optimization, another area where OSPF is in need for external control, we found that decision making has the chance to be more effective at higher levels of the network management hierarchy, due to the greater amount of knowledge that can be made available to the optimization algorithm. For further information on standardizing autonomic behaviors, DEs, MEs and their interactions over well-defined control loop hierarchies, see the EFIPSANS project page at [5], or consult [4].

We believe that taking a fresh look at OSPF from the standpoint of autonomic networking, identifying some of its intrinsic self-* functionality and discovering the ways these functions can be enriched, three ends we tackled in this

paper, will help better understand the fundamentals of how protocol-intrinsic and external control-loops need to interact wherever necessary, and it will be instructive in engineering a truly self-adaptive future routing architecture, as the one illustrated in this paper.

Acknowledgement

The first author was supported by the Janos Bolyai Fellowship of the Hungarian Academy of Sciences. This work was carried out as part of the EU FP7 EFIP-SANS project [5]. The authors would like to thank for the anonymous reviewers for their insightful comments, which greatly contributed to advance the quality of the paper.

References

1. Moy, J.: OSPF Version 2. RFC 2328 (April 1998)
2. Oran, D.: OSI IS-IS Intra-domain Routing Protocol. RFC 1142 (February 1990)
3. Fortz, B., Thorup, M.: Internet traffic engineering by optimizing OSPF weights. In: INFOCOM (2). (2000) 519–528
4. Chaparadza, R., Papavassiliou, S., Kastrinogiannis, T., Vigoureux, M., Dotaro, E., Davy, A., Quinn, K., Wodczak, M., Toth, A., Liakopoulos, A., Wilson, M.: Creating a viable evolution path towards self-managing future internet via a standardizable reference model for autonomic network engineering. In: Proceedings of FIA'09, also published in the Future Internet Book, Prague (May 2009)
5. The EFIPSANS Project: <http://www.efipsans.org>.
6. Katz, D., Kompella, K., Yeung, D.: Traffic Engineering (TE) Extensions to OSPF Version 2. RFC 3630 (September 2003)
7. Kompella, K., Rekhter, Y.: OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS). RFC 4203 (October 2005)
8. Apostolopoulos, G., Kama, S., Williams, D., Guerin, R., Orda, A., Przygienda, T.: QoS Routing Mechanisms and OSPF Extensions. RFC 2676 (August 1999)
9. Villamizar, C.: OSPF Optimized Multipath (OSPF-OMP). Internet Draft (Feb 1999)
10. Shand, M., Bryant, S.: IP Fast Reroute Framework. Internet Draft (Feb 2009)
11. Katz, D., Ward, D.: Bidirectional forwarding detection. Internet Draft (Feb 2009)
12. Enyedi, G., Szilágyi, P., Rétvári, G., Császár, A.: IP Fast ReRoute: Lightweight Not-Via without additional addresses. to appear at INFOCOM'09 (May 2009)
13. Pei, G., Gerla, M., Chen, T.W.: Fisheye state routing: A routing scheme for ad hoc wireless networks. In: Proceedings of ICC 2000, New Orleans, LA (June 2000)
14. Levchenko, K., Voelker, G.M., Paturi, R., Savage, S.: XL: an efficient network routing algorithm. SIGCOMM Comput. Commun. Rev. **38**(4) (2008) 15–26
15. Khanna, A., Zinky, J.: The revised arpanet routing metric. SIGCOMM Comput. Commun. Rev. **19**(4) (1989) 45–56
16. Rétvári, G., Bíró, J.J., Cinkler, T.: On shortest path representation. IEEE/ACM Trans. Netw. **15**(6) (2007) 1293–1306
17. Thorup, M.: OSPF areas considered harmful. Internet Draft (April 2003)
18. Teixeira, R., Shaikh, A., Griffin, T., Rexford, J.: Dynamics of hot-potato routing in IP networks. SIGMETRICS Perform. Eval. Rev. **32**(1) (2004) 307–319