Extending Routers Utilisability and Life Cycle through Automated Configuration Management

Francisco Rodríguez, Mickaël Hoerdt, and Laurent Mathy

Computing Department, InfoLab21
Lancaster University, Lancaster, UK
{f.rodriguez,m.hoerdt,laurent}@comp.lancs.ac.uk
http://www.comp.lancs.ac.uk

Abstract. We present the design of a distributed router platform aimed at consolidating multiple hardware routers. The goal of the approach is twofold: firstly decouple the logical routing and forwarding functionality from the limitations of the hardware that runs it, through automated configuration management only; and secondly, give component routers a longer lease of life, as constituting parts of a larger router system. We focus on the logical intra-domain routing function provided by routers, and show the need for a centralized intra-domain route server.

Key words: Aggregated IP Router, Router Management Automation, Routing Management, Route Server

1 Introduction

Internet Service Providers (ISPs) and enterprise companies have a rather characteristic equipment replacement pattern: new, high performance routers are usually introduced in the core of their network, pushing existing routers towards the network edge, with routers already at the edge being often decomissioned before the end of their hardware life cycle. The basic assumption of this replacement pattern is that routers are usually seen as single role entities, only interacting with each others to create a network of monolithic packet routing devices. There are two reasons for this: firstly most routers manufacturers usually sell their routers without the capabilities to be reprogramed, effectively preventing their owners to perform any structural modification if there was the need to. Secondly, the current ISPs business model is not designed to provide new packet processing abilities at the router level but to manage their network with an acceptable level of service and performance by using the available level of configuration on their routers.

Inline with the previous restrictions, we demonstrate in this paper a technique that has the potential to change the way routers are replaced and managed by ISPs nowadays. We show the feasability of using the routers themselves as the basic building block to compose bigger (logical) routers only by acting at their configuration level. We show that it is possible to build what we called an Aggregated IP Router (AGIR) out of standard router devices without relying on their internal structure. This makes our proposal applicable to most routers. This offers the means to compose and recompose routers out of heterogenous hardware boxes while being independent from any particular implementation.

The main idea behind an Aggregated IP Router (AGIR) is essentially to assemble a single logical router entity from a set of confederated routers regardless its implementation and hardware specifications. A set of confederated routers is automatically configured and managed in order to appear as a single router from an operational perspective. Our goal here is to aggregate routers capabilities under a single automated manageable entity that still serves as a router.

It is important to highlight that an AGIR is not a new router architecture or a technique to split in a predefined fashion router's tasks through a set of hardware components, but a technique to build bigger routers out of a set of heterogenous routers without modifying their internal structure. In a sense, the routers themselves are components of an AGIR. Therefore, all the component routers have to be configured cleverly and automatically in order to complete all the tasks required to provide a single unified router function.

The remainder of the paper is organized as follows. Section 2 describes the concept of Aggregated IP Routers. Section 2.2 presents the challenges that need to be solved for building them. Section 3 describes the proposed design for providing our Aggregated IP Routers with intra-domain routing. Section 4 describes our implementation and evaluation in which we use as a case of study the Open Short Path First (OSPF) protocol. Section 5 provides an overview of the related work. Finally, in section 6, we highlight our conclusions and give directions for future work.

2 Aggregated IP Routers

In essence, AGIRs resemble distributed routers and, as them, offer the same advantages derived from a component based architecture [2–4], but with a *significant* variation. Instead of designing hardware boxes with specific software, tasks, and new protocols, we take advantage of the already existing technology, uniquely relying on the basic configuration interface provided by all routers. By doing this, we investigate the level of flexibility that current routers can offer when composed with each other at the configuration level in order to provide the same functionalities of a router.

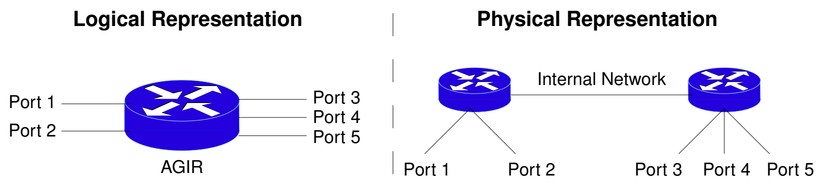


Fig. 1. Aggregated IP Router Abstraction

These routers are interconnected and can interact between each other uniquely at the IP level. Such design seems to appear as a network of routers, but is actually one single router from an external point of view, as illustrated in Fig.1. In principle, any routers regardless its architecture, i.e. hardware routers, software routers and virtual routers, could be included in the composition.

2.1 Benefits

AGIRs provided several benefits. For instance, an AGIR can provide functionalities which are not presented on the individual routers, but presented in the aggregated. For example, AGIR provided a higher level of availability by being composed by multiple routers. Likewise, it might appeared the case that for enabling a certain application which is not currently presented in one router, e.g. NAT or IPSec, can be enabled in the aggregated just by adding a router with such capability without having to completely exchange the router. Additionally, they have a considerably extended life cycle as more component just need to be added for increasing its capacity. Furthermore, since their components are essentially seen as black boxes, only accessible via their configuration interface, they are independent from any router implementation. Moreover, they offer an increased level of flexibility, reliability and scalability in comparison with single chassis routers due to their distributed architecture nature.

2.2 Challenges

Building a AGIR out of closed router boxes[1] poses many challenges, since routers are designed merely to interact between each other at the network layer (IP) in order to build networks, **not** to compose other routers.

Internal networking: Since AGIRs are integrated by N routers residing in N hardware boxes, also referred as AGIR members, they require to be interconnected for communicating between each other. This basic requirement can be fulfilled by using a high speed switch. This switch will provide the means to create an internal network, namely Forwarding Private Channel (FPC). This FPC represents the equivalent to a router's backplane. It is important to highlight that having such internal network introduces new failures scenarios that currently routers do not consider, e.g. the partial failure of the internal network due to a switch's port failure or a member's port failure.

Port awareness: All AGIR's members lack of a complete AGIR's ports awarness as well as a lack of forwarding paths port awareness to all other members. AGIR's members need to have a complete knowledge of all the ports which are contained in the AGIR as well as the forwarding path to reach them.

Heterogenous configurations: Another important challenge to solve is to define a way to represent AGIR's configuration. AGIRs are integrated by multiple members, so it is crucial to have a way to represent and identify them. Furthermore, it is important to create adequate reconfiguration policies for AGIR's

[1] Any router regardless its type and make capable of performing routing and forwarding functions.

members in order to reflect AGIR configuration changes. Moreover, AGIR's members might require configuration changes during its operation time; changes that required to be reflected in all AGIR's members.

Unified Routing: One of the principal challenges for building AGIRs is to provide them with one of router's main capabilities: routing. This capability represents routers' capacity for computing the route paths which they use for forwarding IP packets. Routers can compute their route paths using two different methods: static routing and dynamic routing.

Static routing or static routes are entries in routers configuration that contain well-known establish route paths. These route paths are called static because they do not change except by manual configuration. Due to its persistent state and implementation simplicity, providing AGIRs with static routing capabilities does not present a problem.

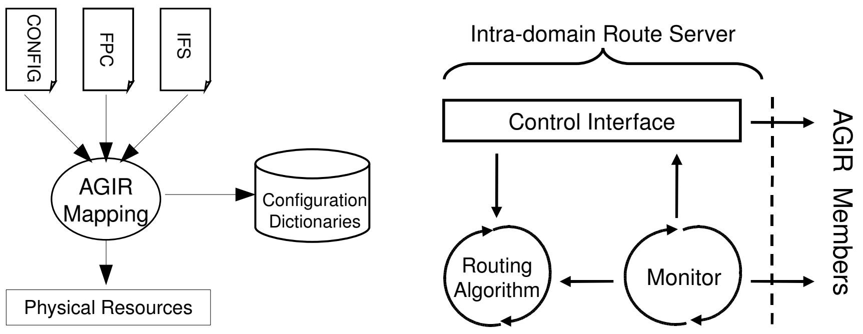
Dynamic routing protocols change their routing decisions to reflect changes in the network topology. In contrast with static routing, providing AGIRs with dynamic routing involves to adjust route paths constantly on the AGIR members ensuring they are provided consistently with respect to AGIR's state at all times. The differences between dynamic routing protocols rely in the method for getting their information, the policies utilised for changing the routes paths and its optimization metric. These routing protocols can be classified based on where they are used in two: inter and intra-domain.

Inter-domain routing protocols are designed for using between Autonomous Systems (ASes). They exchange routing information at the AS level. This means that they calculate the forwarding path in terms of number of ASes, and do not calculate the route through individual router hops within an AS. Currently, inter-domain routing protocols count with multiple solutions that allow them to summarize the view of a set of routers and make them appear as a single router between ASes, such as BGP Route Server and iBGP configuration. Providing AGIRs with an inter-domain routing protocol, such as BGP, does not present subtantial difficulties. The nature of the inter-domain routing protocols works in favour of building AGIRs.

Intra-domain routing protocols are used within an AS or routing domain. They calculate their paths based on router hops within an AS. These routing protocols keep an exact track of how to get from one destination to another inside a network or set of networks that are under the same AS management. An important characteristic of some intra-domain routing protocols, e.g. OSPF and IS-IS, is that they defined a hierarchical routing scheme. This scheme refers to the ability to federate a set of routers in order to segment a large routing domain into "subdomains". The motives are to ease the management complexity of such large routing domains as well as to provide traffic engineering capabilities. These subdomains have a different routing computing policy within them and between them. Learned routes within the subdomain are preferred over other learned routes. In contrast, introducing a new router to an existing domain does not add another hierarchical level of routing as it is done when introducing a new subdomain.

3 Designing an Aggregated IP Router

The proposed AGIR's design is driven by the challenges mentioned in section 2.2. The construction of an AGIR includes: 1) mapping the AGIR to its AGIR's members resources, 2) computing the routes of the AGIR members, and 3) monitoring AGIR's state. Following, we discuss separately each one of these steps in the remainder of the section.



(a) AGIR's Representation (b) Route Server Abstraction

Fig. 2. AGIR's Design

3.1 Mapping the AGIR's members resources

For tackling AGIR's representation problem, we organized the AGIR's information in three different repositories. These files, shown in Fig.2(a), store all the necessary information for assembling a AGIR. User's AGIR's configuration file is represented by the **Config** repository. This repository contains a simplified view of AGIR and stores essential information for mapping correctly to the hardware routers the configuration settings of the user. These settings covers all the information related to AGIR's ports, such as IP information, as well as the routing protocols settings and other applications, e.g. NAT, packet filtering, etc.

AGIR's internal network is represented by the **FPC** repository. This repository contains all the necessary data for setting the communication channels betwen AGIR members. The data stored in this respository describes how an AGIR is composed by listing all its members' ports with its corresponding address within AGIR's internal network.

For solving the port awareness problem, we define the **IFS** repository. This repository stores the location and description of all AGIR's ports. Using the information in IFS, we propose the utilization of static routes for specifying AGIR's ports in every AGIR's member. In this way, AGIR's members will be aware of all the ports that composed the AGIR to which they belong. We have chosen to use static routes in our AGIR design due to the flexibility that they offer for mapping routes at the configuration level.

As AGIR members are accessed in a closed box approach, we need a way to translate their various configuration settings into a unified AGIR configuration. This is represented in our AGIR design by a collection of configuration dictionaries. These are used to set and modify the different configuration settings needed in AGIR members for representing AGIR's configuration settings.

3.2 Computing Aggregated IP Routers Members Routes

In order to overcome the limitation that intra-domain routing protocols present for implementing them in AGIRs, we propose the use of a centralized route server (RS). The main idea of having a RS for AGIRs is to centralize the view of every physical router (AGIR member) participating in the intra-domain routing protocol. This view can be shared accordingly with all AGIR members without having to compute more than once the intra-domain routing protocol, easing the routing management for such setup.

Since the RS acts as AGIR's intra-domain routing engine, the AGIR's RS will need to perform the same tasks that routers perform when running an intra-domain routing protocol, i.e. receive and transmit routing packets, calculate AGIR's routes, calculate and install AGIR members correspondant forwarding tables, as well as to guarantee consistent intra-domain routing announcing and AGIR members forwarding tables consistent with AGIR's real state. We have splitted and grouped the previous tasks, as shown in Fig.2(b), through three different processes: intra-domain routing algorithm, monitor, and configuration policies.

The **routing algorithm** process represents the core of our RS design. This process is in charge of computing AGIR's intra-domain routes by using the corresponding algorithm. The algorithm to run by the RS would depend on the configured routing protocol, e.g. Djiskstra algorithm for OSPF. Furthermore, it is in charge of receiving the incoming routing packets through AGIR's members. Likewise, it is responsible for generating and sending the corresponding routing packets.

One of the main issues, as well as advantages, of using a distributed architecture in routers is that they are integrated by multiple devices. This helps to improve routers' resiliency at the time that it decreases its failure probability[1]. In the AGIR's case, not only the later is true but also introduces a new problem to its RS. Since all AGIR components are disseminated through different hardware boxes, AGIR's RS has to monitor them for detecting any failure and consequently computed correctly AGIR's OSPF routing and every AGIR member forwarding table as well as announcing AGIR's real state to its neighbours. RS's **monitor** process assignment are the prompt detection of routing changes as well as the prompt detection and location of AGIR's failures. Under these circumstances, it will turn on the respective alarm.

As its name suggests, the **control interface** is RS's interface for interacting with its intra-domain routing algorithm process as well as with AGIR members.

This interface contains the policies that need to be applied for modifying AGIR members forwarding tables and/or the intra-domain routing algorithm process behaviour based on the type of alarm received from the monitor process.

4 Implementation and Evaluation

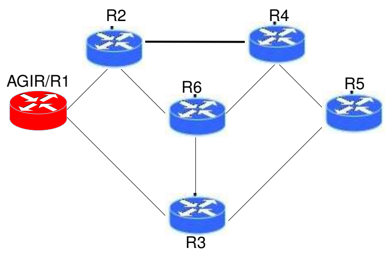
In order to show the feasability of the proposed AGIRs, we have implemented a AGIR mapper and a AGIR intra-domain RS. The AGIR mapper has been developed using Python programming language. Its main purpose is to map AGIR's configuration settings into AGIR's members based on the three different repositories, described in section 3. The commands used by the AGIR mapper for configuring the routers are included in the implemented configuration dictionaries. At the moment, we have implemented only the configuration dictionaries for IOS based Cisco Systems routers and PCs running Linux Debian OS.

For AGIR's RS implementation we have developed its monitor and control interface using Python programming language; as for the routing algorithm we have only implemented Djiskstra algorithm for supporting the OSPF routing protocol. For implementing this routing algorithm, we have used an adapted version of XORP[13] software.

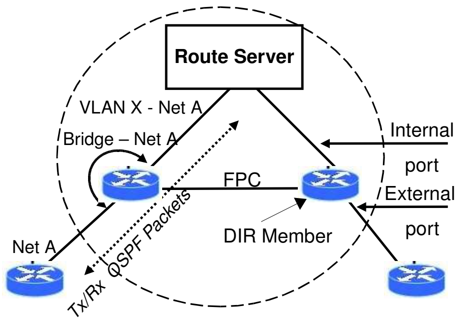
The first problem that we have faced while running our experiments was to cross the OSPF packets through the AGIR members. The reason behind this problem is OSPF packet's TTL value of 1[11]. This value prevents routers to further forward such packets. For overcoming this problem, we have configured AGIR members as bridge-routers[12]. In this way, OSPF packets can cross through AGIR members and all other packets are routed based on AGIR member's forwarding table, as illustrated in Fig.3(b).

Our evaluation tests have been carried out in a virtual environment. Therefore, we have emulated the network topology, shown in Fig.3(a), using Kernel-base Virtual Machine (KVM) software running in two Dell PowerEdge 1950 servers with 2 64-bit Quad-Core Intel Xeon processors. Our evaluation methodology has been to first run tests in our virtual environment using uniquely routers, XORP software running in Linux Debian 2.6.22-3 virtual machines (vms), generating a reference point to compare against the introduction of one AGIR in the same network topology. For composing AGIRs, we have used Linux Debian 2.6.22-3 vms for all its components. For this study, we have used an AGIR composed by two members and one RS, as the one illustrated in Fig.3(b). We have defined in our emulated network, shown in Fig.3(a), R1 as the device to observe through our experiments, and consequently its neigbhours R2 and R2. For AGIR's studies, we have replaced R1 for a AGIR.

We have studied two different aspects of AGIRs using our implementation. First, we have measured the impact of using AGIRs in OSPF networks. And second, we have evaluated RS's performance during all the possible failures that AGIRs could present, ensuring that it provides AGIR with consistent intra-domain routing.



(a) Network Topology



(b) Aggregated IP Router

Fig. 3. Testbed

4.1 Impact in OSPF Networks

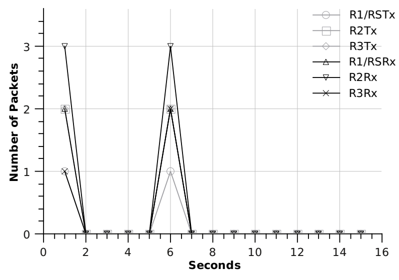
In order to measure the impact of using AGIRs in an OSPF network, we have studied at its behaviour in the event of a failure within the network. In this context, we have simulated in R3 a failure in its port connected to R6 while capturing the generated link state advertisement (LSA) packets due to the failure. We ran the experiment several times and plot the average number of observed LSA packets during the failure and its time distribution.

The results, plotted in Fig.4(a), using and without using an AGIR where the same. When R3's port failure occurs, R3 detects it and floods its remaining reachable neighbours with an LSA update. This event triggers an SPT recomputation in all routers and an announcement later, after 5 seconds on average, of its new state. Using a RS as AGIR's routing engine do not have any impact in OSPF networks when having to recompute the Shortest Path Tree (SPT) after a network failure. The use of AGIR members as bridge-routers does not introduce significant packet distribution variation in the reception and in the transmission of the LSA packets to and from the RS towards its neighbors.

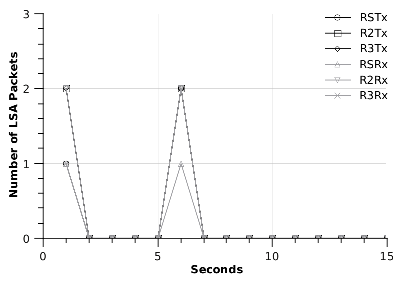
4.2 Providing Consistent Intra-domain Routing

As in section 4.1, we have carried out a comparative study between the different types of failures presented in standalone routers vs AGIR's failures. Using the same methodology of the previous study, we have simulated the different types of failures for both types of routers.

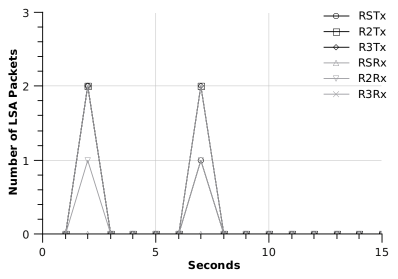
Neighbour's port failure are not failures presented within the AGIR router, but still affect its behaviour. The results from our experiments when simulating this type of failure where the same in the AGIR and the non-AGIR cases. The LSA packet distribution graph, shown in Fig.4, describes OSPF behaviour. Based on results, we can say that AGIRs do not have a negative impact at the routing level. Nevertheless, the utilisation of a RS introduces a natural delay when a SPT recomputation takes place at AGIR's forwarding level. This



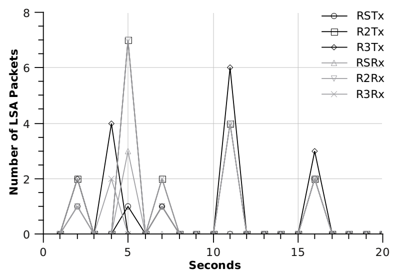
(a) Network Failure Case



(b) Neighbour's Port Case



(c) AGIR Member's Port Case



(d) AGIR's FPC Case

Fig. 4. LSA Packet Distribution

delay is mainly caused by RS's additional tasks: computing and installing the corresponding forwarding tables for every AGIR member.

For determing the mentioned delay, we have measured the reconvergence time for both cases in our emulated enviroment by running the following experiment. On one router (RO), we have generated every 100 ms an ICMP request to a learned OSPF route. Then, we have simulated a port failure in another router (RF) through which the ICMP packets go through to reach its final destination. In this way we defined router's convergence time as: $\Delta T = T2 - T1$; where T1 is the time at which RO stopped receiving ICMP replies and T2 the time at which RO started receiving ICMP replies after the failure.

We ran several times the previous experiment and obtained that the average OSPF convergence time given by a standalone router, in our emulated enviroment, is 3.1 seconds. For AGIR's case the convergence time was significantly above previous case, about 1.2 seconds more. Analyzing our RS implementation, we realized that waiting for the OSPF process to converge before starting to compute AGIR members forwarding table was too costly. Consequently, we decided to integrate into our RS prototype a routine which precomputes AGIR members' forwarding tables in the event of a failure within the AGIR. By doing so, RS OSPF's convergence time when having a failure is composed as follows: $RSCT = DT + IT$; where DT is the time for RS to detect the existance of a failure and IT the time for the RS to install in AGIR members the corresponding forwarding updates. By using this pre-computation technique, we have reduced the average DT for a neighbours port failures to 1.3 seconds while the average IT was 1.2 seconds. So on average, our RS takes 2.5 seconds to update AGIR members' forwarding table.

AGIR's port failures can be classified into two different types of failures: those happening on the **RS's ports** and those happening on the **AGIR member's ports**. Although both failures are the equivalent of having a router's port failure they do not have the same effects in RS behaviour; therefore we have analyzed them separately.

The LSA packets distribution for router's port failure and RS's port failure are fairly the same. The results from both cases present the same behaviour as the one for router's neighbour case, illustrated in Fig.4. The reason for this is that the failure, in both cases, occurred in the physical device running OSPF. During a RS's port failure its very important the DT component, since this will trigger the action for updating AGIR members forwarding table. We have measured such time and obtained that on average the DT for this type of failure is 1.4 seconds.

From the results, shown in Fig.4(c), we realized that in the event of a **AGIR member's port failure**, the LSA packet distribution suffers a shift to the right in comparison to all previous cases. This shift to the right represents RS's DT since the failure did not occur in the device where the routing computations takes place. According to our measurements, DT for this failure type is on average about 1.1 seconds.

A **failure in AGIR's internal network** is the equivalent to a failure in router's backplane. In standalone routers this failure will completely halt its packet forwarding capabilities. In contrast, such failure in AGIRs would only prevent AGIR members to forward packets between them but allowing AGIRs RS to keep exchanging routing information with its OSPF neighbours; consequently, generating routing and fowrading inconsistency in AGIR. For avoiding the latter, we have designed and implemented in our RS a solution for dealing with this very particular case. This solution consists in splitting the OSPF route calculation in the same number of OSPF processes in which AGIR has been splitted by the failure.

We have simulated several times a failure in AGIR's internal network and noted from results, shown in Fig.4(d), that there is a delay in announcing the failure caused by DT. As in all previous failure cases, we have use the approach of precomputing AGIR members' forwarding table for avoiding to wait for OSPF to reconverge which on average can take up to 11 seconds. DT for this type of failure, based on our measures, is on average 1.2 seconds.

5 Related Work

The use of single logical router entities for representing a set of confederated routers to provide a service is not a new concept in networking; it has being widely used with different purposes. Digital was one of the pioneers using and developing this concept in [5]. Its original idea of providing a fast failover mechanism for IP routers or line failures rapidly evolved and converted into what we know nowadays as VRRP. VRRP used logical routers as a way of monitoring a router in case of a failure providing a backup method for the network [6]. The reasons behind VRRP differs from ours, namely to consolidate multiple heterogenous hardware boxes conforming a single logical router.

Another work related to the use of logical router entities is what we know as virtual routers [7, 8]. The main aim of virtual routers is to maximize the hardware resources utilization of high-end routers. As mentioned in [7], virtual routers not only allow routers to maximize hardware resources, but also provide a flexible platform to offer a broad diversity of services e.g. MPLS VPNs[8]. These virtual routers differ from our main purpose inversely. While we are trying to consolidate a set of hardware routers to conform one router, virtual routers try to split the hardware resources of a hardware router in N virtual routers. Furthermore, virtual routers running in commercial platforms are usually limited to reside on a single hardware box. Moreover, AGIRs provides the same functionality obtained by deploying a router, situation that differs in MPLS VPNs case since some of the functionalities provided by the router, e.g. routing, are entirely entitled to the ISPs.

Another benefit that logical routers have proved to provide is the one exposed in [9, 10]. Both works aim to propose an architecture for decreasing network downtime during network maintenance by using cooperatively ISP's infrastructure. They remove the static binding that customers have with ISP's hardware, looking at customers more as a service and less as another hardware box to interconnect. They achieved the later by providing to ISP's customers logical routers and not hardware routers as done nowadays. Their purpose and the way they conceived logical routers, in a single hardware box, differs from our main aim and approach for providing AGIRs.

ForCES framework [2] is the closest work in comparison to our AGIR proposal. In a sense, both works try to provide the means for composing routers from multiple hardware boxes, but differing very much in their proposed method.While ForCES defines an architecture for creating a protocol that allows the utilization of heterogenous NE's components, i.e. from different vendors, for conforming a router, AGIR solution considers a different method, namely to manipulate the routers in a clever and automatic fashion. ForCES objectives follow the line to enable its idea by incorporating a new IPC protocol into the hardware boxes, while our work suggests a new and more effective router's configuration management method. It is also important to highlight, that AGIR considers all types of routers regardless their architecture, so eventually routers built under ForCES framework could be utilised for conforming AGIRs by just adding its corresponding configuration dictionary.

6 Conclusions and Future Work

In this paper, we have described the design of an aggregated IP router platform, aimed at consolidating a set of heterogenous hardware routers connected by a high-speed local interconnect. We have mainly focused on the issues associated to the support of a single logical intra-domain OSPF entity on such platform.

We have analyzed the different failures scenarios in which such platform might appeared and seen, that in most of the cases, it provides an acceptable level of performance. We observed that such platform posses a high level of availability and flexibility, which are presented when using a set of routers in contrast with individual routers performance. Taking the latter in consideration, we believe that the flexibility brought by our aggregated IP routers platform, along with the extension of routers' life-cycle, open up the prospect of various new applications for the platform.

References

1. R. Ramjee and F. Ansari and M. Havemann and T. V. Lakshman and T. Nandagopal and K. K. Sabnani and T. Y. C. Woo. Separating control software from routers, First International Conference on Communication Systems Software and Middleware, New Delhi, India, Jan. 2006.
2. L. Yang and R. Dantu and T. Anderson and R. Gopal Forwarding and Control Element Separation (ForCES) Framework, RFC 3746, Apr. 2004.
3. O. Hagsand1 and M. Hidell and P. Sjdin. Design and Implementation of a Distributed Router, IEEE International Symposium on Signal Processing and Information Technology, Athens, Greece, December 2005.
4. M. Hidell and P. Sjdin and O. Hagsand Router Architectures Tutorial at Networking 2004, Athens, Greece, May 2004.
5. P. L. Higginson and M. C. Shand Development of router clusters to provide fast failover in IP networks, Digital Tech. J. Vol.3, Issue 9, Acton, MA., USA, Jan. 1997.
6. R. Hinden Virtual Router Redundancy Protocol (VRRP), RFC 3768, April 2004.
7. Juniper Networks. Intelligent Logical Router Service, White Paper, Oct. 2004.
8. Cisco Systems. Introduction to Cisco MPLS VPN Technology, www.cisco.com
9. M. Agrawal and S. R. Bailey and A. Greenberg and J. Pastor and P. Sebos and S. Seshan and K. van der Merwe and J. Yates. RouterFarm: towards a dynamic, manageable network edge, INM '06: Proceedings of the 2006 SIGCOMM workshop on Internet network management, Pisa, Italy, Sept. 2006.
10. Y. Wang and E. Keller and B. Biskeborn and J. van der Merwe and J. Rexford. Virtual routers on the move: live router migration as a network-management primitive, SIGCOMM Comput. Commun. Rev. Vol. 38, Issue 4, New York, NY, USA, Oct. 2008.
11. J. Moy. OSPF: Anatomy of an Internet Routing Protocol, Addison-Wesley, Jan. 1998.
12. Cisco Systems. Understanding and Configuring VLAN Routing and Bridging on a Router Using the IRB Feature, www.cisco.com
13. M. Handley and E. Kohler and A. Ghosh and O. Hodson and P. Radoslavov. Designing Extensible IP Router Software, In the proceedings of the 2nd USENIX Symposium on Networked Systems Design and Implementation, Boston, Massachusetts, USA, May 2005.