

# Routing Engine Architecture for Next Generation Routers: Evolutional Trends

Kim-Khoa Nguyen (Corresponding author)

Department of Electrical and Computer Engineering, University of Sherbrooke

2500 boul. Université, Sherbrooke, Quebec J1K 2R1, Canada

Tel: 1-514-848-2424#7062 E-mail: [kk\\_nguye@encs.concordia.ca](mailto:kk_nguye@encs.concordia.ca)

Brigitte Jaumard

Concordia Institute for Information Systems, Concordia University

1515 St. Catherine W., Montreal, Quebec H3G 2W1, Canada

Tel: 1-514-848-2424#5380 E-mail: [bjaumard@ciise.concordia.ca](mailto:bjaumard@ciise.concordia.ca)

## Abstract

The routing engine is the essential part of a router. As a software component, the routing engine is used to control the router activities and to build the data forwarding table. Along with the hardware evolution, several software generations for routing engines have been experienced. In this paper, we first review the different routing engine architectures over the time and their main components. In some previous papers, we have investigated new directions for routing engine development for next generation routers, where specific distributed architectures have been proposed for several routing protocols. Taking into account the increasing traffic in the core networks, we next propose a framework integrating all specific distributed routing protocol architectures in order to significantly improve the scalability of the next generation routers. Bottlenecks are reduced, resulting in improving both the overall performance and the resiliency in the presence of faults. The scalability of the proposed distributed framework is estimated with respect to each routing protocol.

**Keywords:** Routing engine, Control plane, Next generation routers, Routing protocols.

## 1. Introduction

New network services have been proposed in the past few years, such as packet tagging, application-level proxies, application-specific packet dropping, performance monitoring, intrusion detection, and assorted filters and firewalls. Besides, scalability, resiliency and availability are emerging for data transmission networks. Router architectures are therefore evolving in order to provide services with the expected requirements. Although software can be easily modified, rebuilding or reprogramming a router to support a new service remains a challenging endeavor. It is also important that the router software has an architecture that may be extended and built on.

Router architectures have experienced three generations [1]. Current routers are built with a crossbar switch fabric which can provide high speed switching capacity. In order to interconnect the routers in the core network so as to deal with the growth of Internet traffic, two approaches can be used. The traditional one is to add more routers, often mid-size routers, into the operator's network, and to form clusters of routers. The newer approach, which has been proposed in [1][2], consists in replacing clusters of routers by very large routers. Critical issues for the first approach include extra cost for management and maintenance, overloading of the routers due to the explosion of the number of control messages and the size of the routing tables, and inefficiency in exploiting high-speed optical interfaces. The second approach, on the other hand, faces the redesign of hardware and software components, such as the switch fabric, control and line cards. Both approaches need either a new software to control and manage geographically distributed routers or a very highly scalable router.

Router software products are often designed for specific router platforms. For example, Cisco IOS cannot be used for Juniper routers. However, current router software shares some common characteristics like modularization, control and data plane separation and management. In this paper, we review and analyze the current software architectures for a router control plane, and then present some new promising directions in architecture development.

Basically, the software architecture of a router is composed of three interdependent functional blocks connected by interfaces:

- *Data plane* (or forwarding plane). It forwards data packets based on longest-prefix match that identifies the outgoing link for each packet, as well as on the access control list (ACLs) that filters packets based on their header fields. Functions such as tunneling, queuing management, and packet scheduling are also implemented therein. The data plane is built in programmable network processors.
- *Control plane*. It runs signaling and routing protocols, and is also responsible for managing routing tables. The control plane computes the forwarding table that is used by the data plane.
- *Management plane*. It handles network management applications, protocol policies, QoS, etc. It also stores and analyzes measurement data.

In legacy routers, and in nearly all software based routers [3], the three planes share the same processing units, thus limiting the router performance. Following the hardware evolution, routers become more scalable with higher switching capacity. Therefore, an independent data plane is recommended in current router architectures. Some recent research [4] proposed to further separate the three planes, hence to increase the overall switching and routing capacity.

This paper first discusses the evolution of the router control plane software, and focuses on the routing engine with respect to the following components: i) Software architecture of the most commonly used routing and signaling protocols: RIP, OSPF, IS-IS, BGP, RSVP, LDP, MPLS, ii) Routing Table Manager (RTM), iii) IP stack, iv) Link manager, and v) Embedded intermediate layer for the cooperation between the protocols and the router hardware components.

We are interested in architectures for large scale routers to be deployed in core networks, such as [5][6]. Throughout this paper, we first discuss different implementation models for each component in various router products. We also develop an integration framework of several distributed architectures we recently proposed for specific routing protocols.

## 2. Main Components of a Router Control Plane

A router control plane usually includes the components which are next described, as shown in Figure 1.

*IGPs (Internal Gateway Protocols).* They are responsible for establishing routes within an AS (Autonomous System). Several IGPs can run simultaneously on the control plane. Typical IGPs are RIP, OSPF and IS-IS. OSPF is implemented on top of IP stack and IS-IS is located on top of OSI stack. Some routers are designed with only one process for each IGP protocol, others can support multiple processes for each IGP protocol. For example, the GateD routers, e.g., [7], support only one OSPF process per router, while Cisco routers, e.g., [8], can support multiple OSPF processes per router. Legacy routers still provide a RIP module. RIP is actually one of the very first routing protocols in networks, but is currently rarely used due to the backward compatibility.

*EGPs (External Gateway Protocols).* They are used to establish routes over different ASs. One of the most used EGPs is BGP. BGP is running on top of TCP. Routers use a loopback address to communicate with the external world in order to establish BGP sessions. Thus, a BGP session between two neighbor routers can be switched from an interface to another one in case of an interface failure. BGP implementation architectures do not differ much throughout router generation evolution. Basically, the BGP module is hosted by a single CPU process. Some products can support multiple BGP processes per router [9].

*Multicast protocols.* The most used multicast protocol is IGMP (Internet Group Management Protocol). IP routers use IGMP to report their multicast group membership to neighboring routers. Similarly, multicast routers use IGMP to discover which of their hosts belong to multicast groups. Some other specific multicast protocols such as PIM (Protocol-Independent Multicast, to identify other multicast destination routers), DVMRP

(Distance Vector Multicast Routing Protocol, to forward multicast datagrams through a network) can also be implemented.

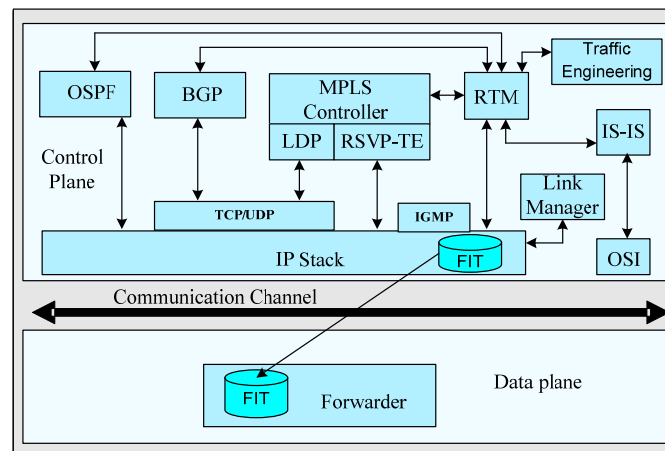


Figure 1: Typical control and data planes and their software components

*RTM (Routing Table Manager)*. The main task of the RTM is to build the Forwarding Information Table (FIT) from one or several routing tables that store the set of static routes and of routes learnt from different routing protocols. As some services can use non-best routes to forward data with respect to user-defined parameters, RTM has to store a whole set of routes and allows users or requested modules to access the route databases and make routing decisions [10]. The RTM dynamically updates the IP stack with the latest revision of the FIT table and keeps it up-to-date whenever changes occur. This is achieved through an interface called routing socket.

*QoS (or Traffic Engineering) module*. It is now required in the current routers, in order to prioritize or differentiate traffic and to ensure the network infrastructure convergence (e.g., voice traffic is carried over data networks).

*IP stack*. It runs the routing protocols and manages the Forwarding Information Table (FIT). It also provides socket services for routing protocols and management applications. In practice, IP stack is usually implemented together with TCP/UDP stack. In addition, IP stack performs regular IP services such as IP header validation, header generation, checksum calculation, packet fragmentation/de-fragmentation, ICMP, etc.

*OSI stack*. It is used primarily to run IS-IS protocol. Its architecture is basically similar to the IP stack.

*Link manager* (so called Interface Process in some router models). Its main role is to manage individual and bundled interfaces, as well as LSP virtual interfaces. An interface can be point-to-point or multi-access. The link manager handles the up/down status as well as the inbound/outbound traffic of each router interface. It is also responsible for traffic filters. In a router with a distributed hardware architecture (i.e., made of several control and line cards interconnected by a crossbar switch fabric), a link manager module is often implemented on each card.

In addition, a specific module dedicated to the chassis management where a chassis

houses a cluster of cards, an independent power supply and a fan assembly, so called chassis process in some router models [6], is also required for chassis-based router architectures. This process is out of the scope of this paper.

An SSH module can also be seen in some router products, e.g., [8], in order to provide remote management services.

When the router hardware is designed with a distributed architecture, the router software involves an intermediate layer to help the cooperation among the protocols and the router hardware components. Such a typical layer, so called Distribution Services (DS) [10], performs the following tasks: i) Manage the set of available cards in the router, ii) Distribute routing software modules on different control cards based on the router configuration, iii) Maintain consistent information with the software distribution, iv) Provide an API to provision new software distribution configuration, and v) Provide a query interface to the distribution API to retrieve software distribution information.

The DS can be seen as a software connector that binds components together and acts as a mediator among them. These modules are based on an OS kernel providing basic operations, such as file system services, memory management, I/O services, etc. An architecture of the DS can be found in [11].

When packets are received by a line module (interface) on the data plane, they are classified based on their ingress interface, MPLS labels, or any field in the packet, using information provided by the control plane. Classification may include traffic queue policies. Actions are performed on the packet as a result of classification, and then packets are forwarded to their egress destination via IP forwarding function. Line module handles packet scheduling, layer-2 encapsulation, and transmission of the packet on the egress network. Weighted-based scheduling could provide relative priorities among queues.

In legacy routers or small scale routers, all routing protocol modules, as well as RTM and stack management modules share the same memory space. They are hosted by a process called routing engine. Recent routers are provided with a modular control plane, which allows the router to have several collaborating routing engines, with some of them used for backup purposes.

### **3. Challenges of Router Control Plane Development**

In [2], the authors have predicted a convergence of router and network architectures, that may lead to a system with several routers and a central processing server. The central processing server performs the routing, traffic engineering and management tasks and updates the forwarding tables of the routers in a given domain or cluster. The routers exclusively perform data forwarding. Such an architecture raises several questions about the size of the routers and of the cluster, as well as about the scalability of the central processing server. If the routers are highly-scalable, they need to be equipped with very high speed interfaces, which are often expensive. In such a case, using these interfaces for intra-domain communications, namely the connections with the central server is not an effective resource utilization. One may think of a finer solution where routers use less expensive interfaces for

communication with the central server, considering that the amount of bandwidth for control messages is much smaller than that used for data switching. However, control message processing might be delayed due to the communications between the central server and the routers, leading to time gaps and potentially service interruptions. An improved solution, presented in [12], is such a system, but where the routers keep the responsibility of some or all control functions, especially the HELLO processing, in order to maintain the router connectivity in case of a central server failure.

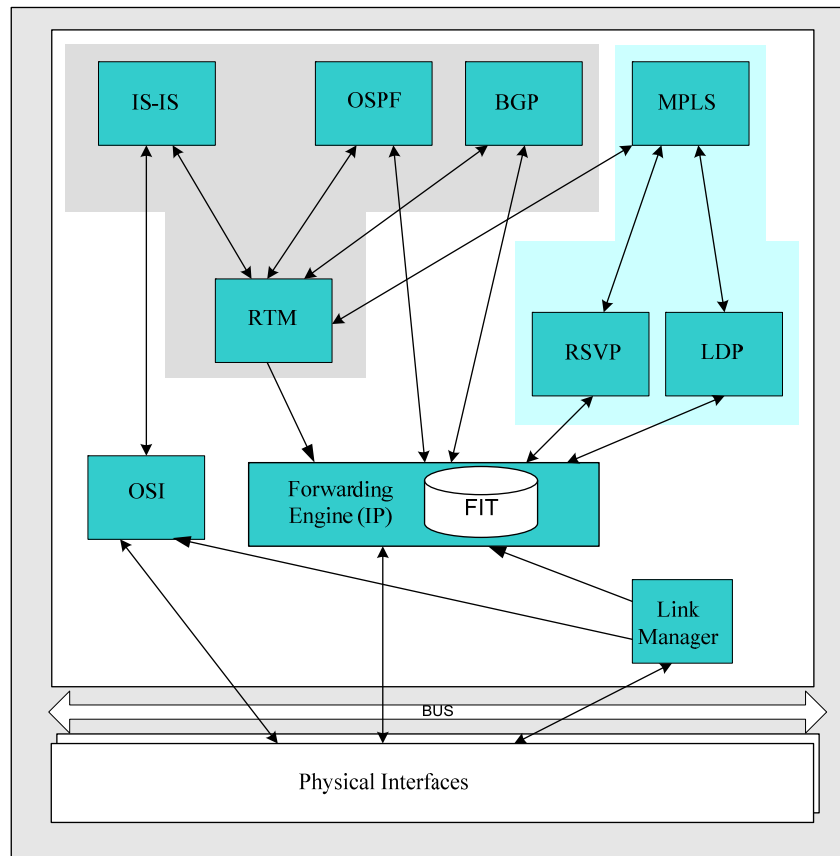


Figure 2: Software Architecture of First and Second Generation Routers

In terms of hardware architecture, legacy routers (i.e., first and second generations) are built with one CPU on a control card handling all basic modules such as routing engine, packet forwarding and service engines. The routing engine handles a set of routing protocols like IS-IS, OSPF, BGP and MPLS that run all together and interchange information such as routes or labels. The exchange and coordination of these protocols are generally done via a Routing Table Manager (RTM). Figure 2 shows the software architecture of a first generation router. In such an architecture, the RTM is responsible for retrieving the information learned from the different routing protocol modules, making decisions for selecting the best routes and generating accordingly the best route table (FIT) which can be used later in forwarding the packets to the corresponding destinations.

The advantage of such an architecture is the ease of management since all the routing protocols run together on the same control card. The synchronization and message exchange mechanisms are also quite simple to implement. However, the critical issue of such legacy

systems is their monolithic code base with all forwarding and routing processes competing for the same CPU and memory resources. Consequently, as the demanding packet forwarding process consumes almost all the CPU capacity, the other functions are left starving CPU cycles. Clearly, this type of software architectures can only be used for small and medium scale routers.

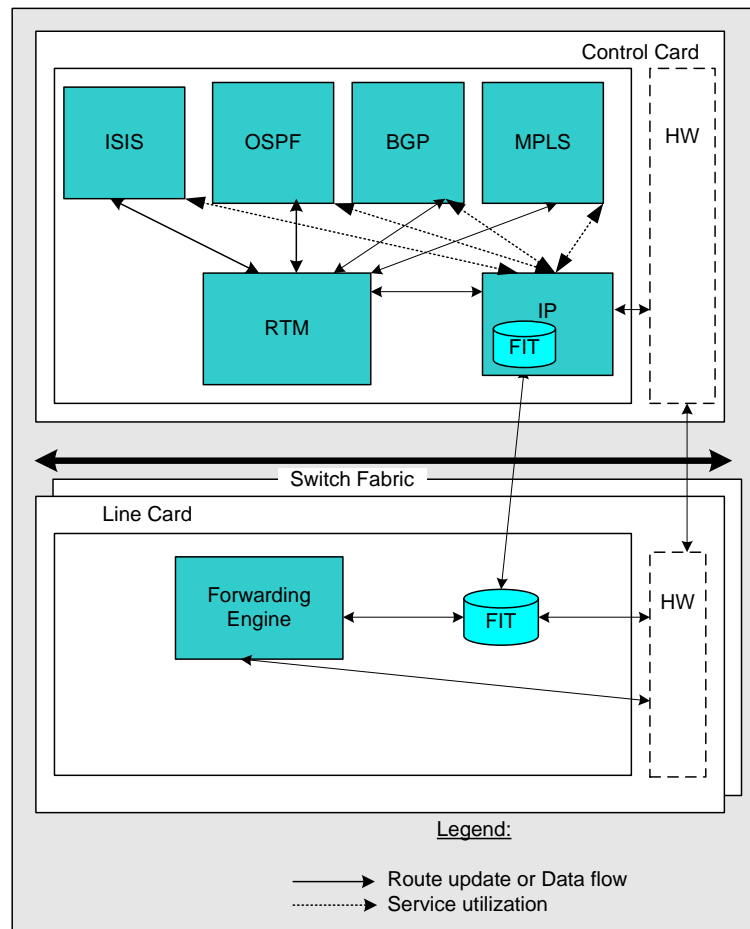


Figure 3: Software Architecture of a current core router

The current generation of routers (i.e., third generation routers) consists of a control card and a set of line cards connected via a switch fabric, i.e., of a distributed hardware platform. Line cards contain very high speed interfaces (i.e., OC-192). Their software architecture is depicted in Figure 3 and consists of:

- One controller card that hosts all routing protocols. There can be an additional control card used for backup and redundancy.
- A given number of line cards which perform: i) IP forwarding and/or MPLS label switching at hardware level, and ii) IP forwarding at software level for exceptional packets (e.g., control packets).

In addition, there is a FIT on each line card. The RTM located on the control card receives the best routes learnt by the routing protocols. Overall best routes are selected and then recorded in the FIT of the IP stack. The FIT on each line card is downloaded from the FIT on



the control card via the switch fabric. The performance and the fault tolerance of the router are hence improved because each line card is able to make the forwarding decisions by itself without the need to send requests to the control card or to use a separate forwarding engine. However, there is still a potential bottleneck at the control card level where all routing protocols run simultaneously. Therefore, such an architecture is still not scalable. The large capacity of the line cards and of the switch fabric is not efficiently exploited due to the location of almost all processor and memory computational resources on the control card.

As the largest vendors in the market, like Cisco and Juniper, do not publish their software architecture, we have little idea about the architecture of their control cards. However, in the recent products, see, e.g., [5] [6], we observed that there is no control function running on the line cards. Therefore, all routing protocols are handled by the control cards. This does not allow the control card to serve a large number of line cards. For example, the current Cisco 12000 series products cannot support more than 16 line cards per chassis [5].

Due to the growth of the Internet routing tables and the web-based traffic, the software architecture used in the current routers (third generation routers) lacks of efficiency and scalability. Third generation routers are moving to the edge of networks and leave room for the next generation routers which need to be more scalable. For example, in 2005, HyperChip Inc. has announced a new core router model which may support a very large number of line cards and control cards (e.g., 64,000) and provide a very high throughput up to 1280 Gbps. The software architecture for next generation routers should therefore be much more distributed in order to be scalable and to take full advantage of the distributed hardware platform entailed by the switch fabric.

#### **4. Trends in Routing Engine Development**

We currently foresee three deployment trends for routing engines in the current router products.

##### *4.1 First Generation: Monolithic Routing Engine*

In a monolithic (or centralized) routing engine, all routing protocol modules are implemented on a control card. They share a common memory space (Figure 4). In a highly scalable control card architecture, several CPUs can be used to host the routing engine through sharing services provided by a multi-tasking OS.



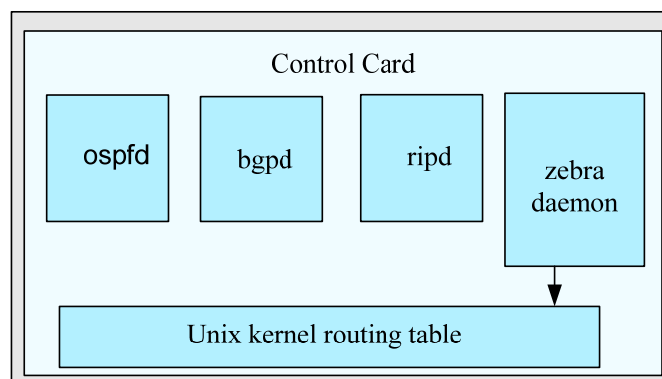


Figure 4: A monolithic routing engine: Example of a Zebra router

In such an architecture, the communication among the different processes is achieved through the OS services (i.e., inter-process communication services). The RTM services are provided by a specific daemon (e.g., Zebra). The processes (i.e., routing protocol modules, stack management modules and RTM) run concurrently on one or a small number of CPUs so the performance is limited. Table access operations are also relatively slow. New protocol modules can be “plugged” to the RTM through defined interfaces.

Figure 4 presents an example of such a monolithic architecture designed by Zebra [9]. It has a separate daemon for each routing protocol implemented that can be run as a stand-alone process within a memory space of a single CPU. There is a specific daemon taking care of the communications between the routing daemons and the kernel routing table. They share the CPU cycles of a single control card. As the RTM is associated with only one routing protocol in use at a time, the performance of such a software architecture is limited. Some slightly more advanced architectures allow the routing daemons to run on different machines and to connect to the RTM through networking protocols. However, the overall performance is not much improved as a congestion point remains at the RTM level. In addition, the data forwarding plane is not designed to handle heavy traffic load. This architecture offers true modularity but lacks extensibility.

#### 4.2 Second Generation: Cluster-based Routing Engine

In a cluster-based architecture, the routing engine modules are distributed on several cards. A typical architecture include a dedicated card for each protocol or so, e.g., one for RTM and MPLS, one for OSPF, one for IS-IS and one for BGP (Figure 5).

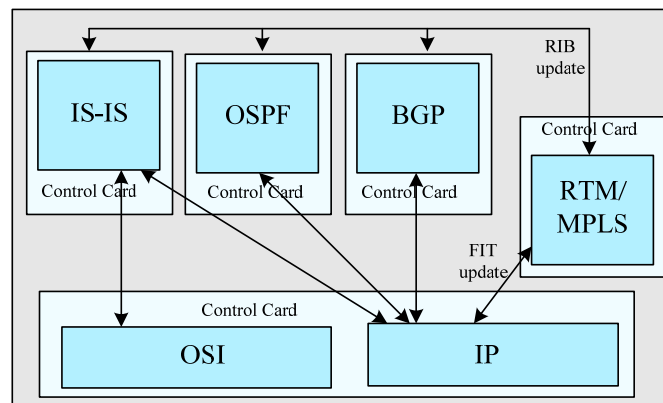


Figure 5: A cluster-based routing engine

A cluster-based routing engine increases the scalability of the router by using more computing and memory resource distributed over several control cards. A Distribution Service (DS) is required in such an architecture to interconnect the different cards, leading to a higher performance of the architecture, compared to the previous one (Section 4.1). However, there are still potential bottlenecks at the RTM card due to the large number of requests performing route lookup and access from all routing protocol modules.

A hierarchical model can also be used to design routing modules, as presented in Figure 6, see, e.g., Juniper [6]. At the very top is a BGP process responsible for creating forwarding table entries. If the BGP process fails, the forwarding table will be flushed, thus disrupting the current traffic. The BGP process relies on lower protocols in the stack, such as OSPF and RSVP for traffic engineering and topology information. If any of these protocols is restarted, BGP and the forwarding plane will also be affected.

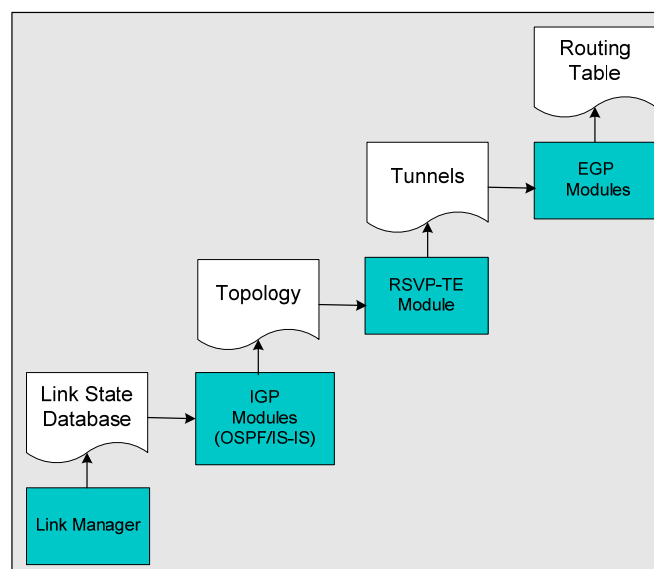


Figure 6: Hierarchical model of routing protocols

#### 4.3 Third generation: Distributed routing engine

A distributed routing engine comprises the modules running on different cards of a router.

As shown in Figure 7, see, e.g., HyperChip as described in [12], each routing protocol module can be divided in two parts, the control component on a control card and the signaling components on the line cards. The signaling component performs protocol I/O operations, such as message exchanges with neighbors and adjacency discovery. The control component is responsible for computing operations, like best route computation, routing access, database synchronization. Such an architecture increases the scalability and resiliency of routing protocols because control card resources are not used for I/O tasks. In addition, a control card failure will not shutdown the routing activity because the current routing sessions are still maintained at the line card level. This improves the router availability at the expense of a memory and computing resource requirement on the line cards. The result is a suitable architecture for next generation routers, as described in [1].

Indeed, in order to accelerate the route update and lookup operations, the RTM is distributed into sub-modules attached to different routing protocols. Each module, so called IGP RTM (e.g., OSPF-RTM or IS-IS-RTM), occupies part of the memory space of the control card. They all run simultaneously on one or several control cards, which reduces considerably the traffic to the routing table. There is a Global RTM (G-RTM) located on a control card that gets routing information from the IGP RTMs in order to update the routing table and consequently the forwarding table of the router.

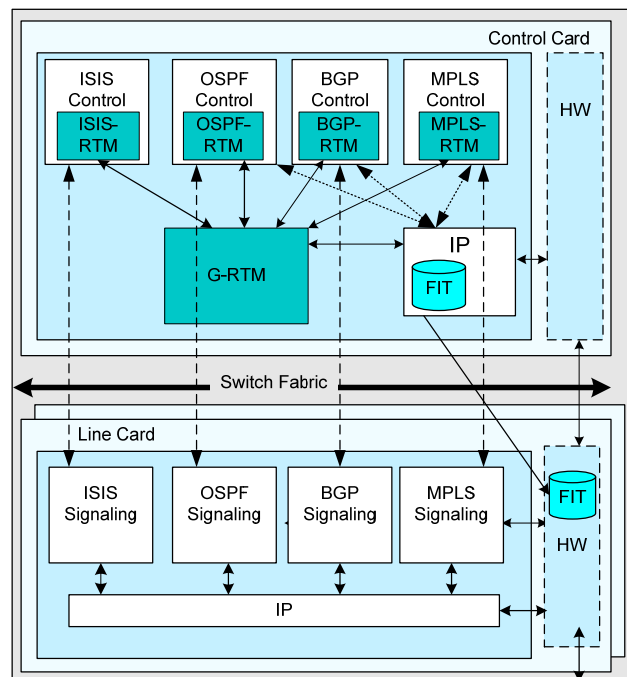


Figure 7: A routing engine with distributed RTM modules

## 5. Control Plane Software Development: Centralized vs. Distributed Architecture

Until very recently, no distributed software architecture for router control plane has been considered. Third generation routers are still provided with no extra memory on the line cards. Therefore control tasks are mainly performed at the control card level. The need of distributed software architectures for routers has led to the born of the next generations. In

Table 1, we compare the control planes developed for widely used routers in the market. Most of them are designed with either monolithic or modular architectures.

Table 1: Comparison of existing control planes

Vendor	Control Plane	Type	OS based
Cisco	IOS	Monolithic	n/a
Cisco	IOS-XE	Monolithic, some modular	IOS, Linux
Cisco	Modular IOS	Monolithic, some modular	IOS, QNX
Cisco	IOS-QNX	Modular	QNX
Alcatel-Lucent	SR-OS	Modular	VxWorks
Redback	SEOS	Modular	NetBSD
Juniper Networks	JUNOS	Modular, some distribution	FreeBSD

The recent research on distributed software is mainly related to the router operating system, which is originally motivated by the trend to extend the forwarding function of a router. The existing architectures, mostly based on open-sources, are aimed primarily at providing a software prototype to implement interfaces or communication protocols among router components. The router prototypes on which the software architectures are developed are of small scale, or even general-purpose computers with no specific chipsets. The software architectures are still implemented with no control function on line cards. More precisely, the line card is considered as a simple forwarding element of the router where no routing, signaling or management task can be hosted.

We classify the current distributed software routers according to process or task distribution:

- *Process distribution.* The router software is composed of independent processes which can run simultaneously on the hardware platform. Each control function (i.e., a routing protocol) is achieved by an independent process. A specific example of such an architecture is Router Plugin [13]. The software framework supports dynamic loading and unloading of plug-in modules at run-time into the kernel of the operating system (OS). Each protocol or forwarding engine is defined as a module. The architecture is implemented in the Net BSD operating system kernel, which is an open source Unix operating system. The forwarding engine is designed with extended functions, such as packet-to-flows mapping and filtering. Another example of distributed software in this category is XORP (eXtensible Open Router Platform) [3] which is also open-source and Unix based. In XORP, the routing software is modularized into one process per protocol and extra processes for management, configuration and coordination. It also defines a Forwarding Engine Abstraction (FEA) which allows running the higher-level subsystem on top of different types of forwarding engines.
- *Task distribution.* Each router function (i.e., routing protocol or forwarding) can be split into different tasks. Each task can be achieved by a router component. Thus functions run at different locations of the router. A typical example of such architectures is ForCES,

presented by IETF [4], which can be considered as the most notable framework for distributed routers. The ForCES architecture is defined in terms of exchanges of information between control elements (CEs) and forwarding elements (FEs). A group of CEs and FEs together forms a network element (NE) that can be considered as a router in the traditional sense. The ForCES protocol is used to coordinate the CEs and FEs. It updates the FEs with configuration information from the CEs, queries for information by the CEs or sends asynchronous event notifications to CEs. Using the ForCES protocol, the CEs may also configure the processing functions on the FEs. Based on the ForCES architecture, one can attempt to redefine the control functions of a router in order to share the processing tasks between the control cards and the line cards. In [14], the authors present a Distributed Control Plane architecture where some message processing, particularly HELLO processing, is handled by the line cards. Therein, an example for distributed OSPF architecture demonstrates that, when HELLO processing is moved to line cards, failures can be detected faster and Shortest Path First (SPF) calculations can be run as frequently as required without affecting the load on the control plane processor.

Basically, process distribution does not efficiently exploit the next-generation router hardware platforms as the processing resource on line cards is not used for control processing. It is rather suitable for third generation routers with multiple control cards.

Task distribution seems more suitable for next generation routers. However, the current ForCES architecture does not take advantage of the hardware features of the next generation routers such as the general purpose CPU and available memory on line cards. As described, a forwarding element in the ForCES architecture may correspond to a network processor (NP) on line cards of a next generation router. Since the network processor is required for key data processing functions, such as table lookup or flow classification, the integration of some control functions into the forwarding element as proposed in [14] may slow down the data forwarding speed of the line card. In addition, the NP is often designed for specific interfaces so reprogramming the NP is costly.

One of the novel approaches we have proposed [10][12] enhances the ForCES framework by exploiting the general-purpose CPU and memory on line cards, instead of relying on the network processor. It keeps the forwarding element intact so that the forwarding performance is not influenced by new control functions implemented on the line cards. In addition, using the general purpose CPU on a line card allows more control functions to be offloaded from the control card in order to increase the scalability.

## **6. A Distributed Control Plane Architecture for Next Generation Routers**

We are now ready to present a framework integrating the distributed architectures previously proposed for specific routing protocols. Taking advantage of the next generation router platform which provides additional processing and memory resources on line cards, we exploited the ability of moving some control functions from control card to line cards. We go one step further along this direction with an integration framework for distributed routing protocol architectures, which is based on the following assumptions:

- The router is based on next-generation architectures, consisting in a distributed platform, with separated control cards and line cards. Line cards are equipped with a plain memory and processing power, and are able to perform all the data forwarding and some control tasks.
- The communication among the router cards is achieved through a specific hardware device (called Switch Fabric, SF) that is able to provide the required bandwidth and other QoS demands. The forecast switching capacity is in the order of a few petabits.
- There is a specific communication channel among line cards, enabling them to exchange control information with a negligible impact on data flows. This channel shares the bandwidth on the switch fabric with data flows. In an implementation environment [11], this channel is designed as an abstract layer called Distributed Service (DS). It provides a synchronization mechanism to manage module activations, monitoring and state transitioning facilities (active, backup, in-service upgrade, etc.). DS maintains a distributed database allowing requesting modules to get appropriate messages. Thus messages to be sent are flooded to DS and the destination will be notified.

Within a distributed control plane framework designed for the control plane of next generation routers, we have proposed a distributed RTM architecture considered as the heart of a router software system [10]. This paper focuses on the routing protocol modules designed in a distributed way, integrated with a distributed RTM. The multicast protocols are kept intact as they depend heavily on the network configurations.

#### *A. BGP*

A BGP architecture may have several stages. Routes come in from a BGP peer and go through an incoming filter bank into the decision process. Combining with the IGP routes from routing tables, best routes are selected and sent out through an outgoing filter bank to peer routers. As the Internet becomes larger and more complex, BGP is likely to manage and maintain a very large number of sessions and routes, (which may reach up to 2.5 millions in some ASs); this growth makes it very hard for a single hardware to maintain the BGP protocol working properly. Hence, a distribution processing scheme for the BGP protocol is needed. The principle for distributing BGP and to make it more scalable is to enable the protocol to handle more BGP sessions with its peers.

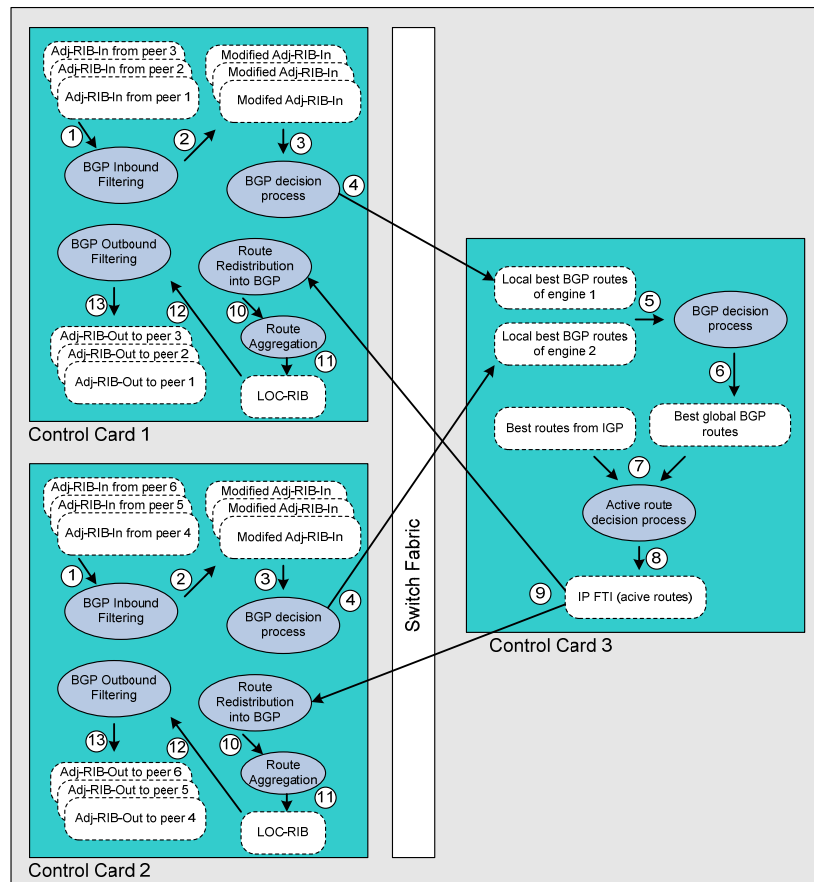


Figure 8: Distributed architecture of BGP

In the proposed architecture [16], the information about the BGP peers and the ones learned from peers, Adj-RIB-In database, is distributed on different control cards. Therefore, each control card is responsible for a limited number of peers and their route information. In a typical configuration, a router in the core network may have three control cards, each one hosts an independent BGP process and each BGP process handles 100-200 adjacencies. Figure 8 shows a router configuration with three control cards where two control cards are used to execute two BGP processes and the third one handles the global routing table. The numbers represent steps in the route processing procedure in order to update the Route Information Base (RIB). Each BGP process running on a control card can be considered as an independent BGP speaker and is responsible for handling a set of BGP peering connections, receiving and storing routes from the corresponding peer subset. The peers from the same autonomous system (AS) are stored on a control card in order to make it easy to run the local policy. Each BGP process runs the in-bound policies and out-bound policies and makes its local decision on the selected best routes. These selected routes are flushed to the L-RTM (Local RTM), and then are sent to G-RTM (Global RTM) where they are processed to select the best routes that are used to create the routing table. These best routes are sent back to all the protocol processes including the BGP ones which are running on the other control cards, and which are participating in the update of those best routes. Thus the other routing protocols on the other control cards, including the BGP protocol, will update their local tables



with the best routes chosen by the G-RTM.

### B. OSPF

The proposed OSPF architecture addresses the scaling issue by distributing the software over multiple control card and line card boards. This architecture has been presented in [15] and now is described in a unified whole framework context where all routing protocols run together. The architecture consists of two main components (Figure 9):

- The OSPF Control Module (OCM) that runs on the control card. It implements a fast processing SPF algorithm to reduce route calculation time (and therefore improves network convergence time). It can also calculate alternate next hops to achieve fast-reroute on link failure. The module can be distributed over multiple control cards when memory usage reaches a single card limit.
- The OSPF Line Module (OLM) that runs on the line card. It allows parallel processing of incoming and outgoing OSPF packets over multiple line cards.

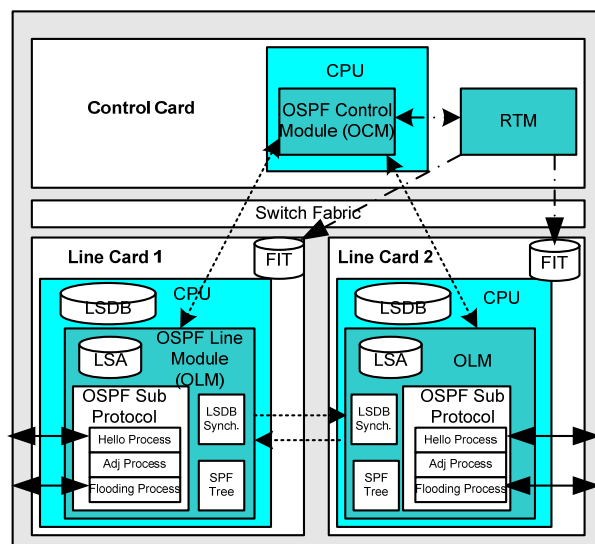


Figure 9: Overview of the proposed distributed OSPF architecture

Generally, both the OCM and the OLM should have a backup in order to achieve the full OSPF resiliency. However, as the information loss of an OLM is considered much less important compared to the loss of the OCM, OLM resiliency is not supported in the current proposed architecture. When an OLM fails, OSPF routing calculation can still take place for the remaining OLMs. However, when the OCM fails, all OSPF routing activities are disrupted for all interfaces and the system must be restarted. Thus, resiliency at the OLM level will be part of our future work.

OCM resiliency support is provided using a backup OCM running on a different control card. If the active OCM fails, the backup OCM will take over the task seamlessly so the loss of the OCM is transparent to external routers.

On the other hand, if an OLM fails, the problem will be detected on the line card and the

OLM software restarted or the card rebooted. Failure on an OLM will be detected by external routers and new adjacency must be reestablished for the failed interface.

Each instance of the OCM can be independently upgraded; there is no requirement for the two instances to be upgraded at the same time. An upgrade of an instance is accomplished by stopping that instance, upgrading it, then starting the instance again. Stopping the active instance causes the backup instance to automatically become active.

The LSA database managed by the OLM, as well as the LSDB managed by the OCM, are provided with five main functions: LSA-Lookup, LSA-Create, LSA-Update, LSA-Delete and LSA-Flush, that are used respectively to lookup and verify a new LSA, to create a new LSA, to update an LSA, to delete an existing LSA and to handle an LSA flushed from the network.

In the proposed distributed architecture, each OLM running on a line card contains a copy of the global LSDB maintained by the OCM on the control card. When a line card is started, the OCM sends OSPF configuration messages to the corresponding OLM, which allows the OLM to run Hello protocol and perform message exchanges according to user settings. From time to time, the OCM sends a request to each OLM in order to obtain some statistical information, such as interface status, performance count, etc.

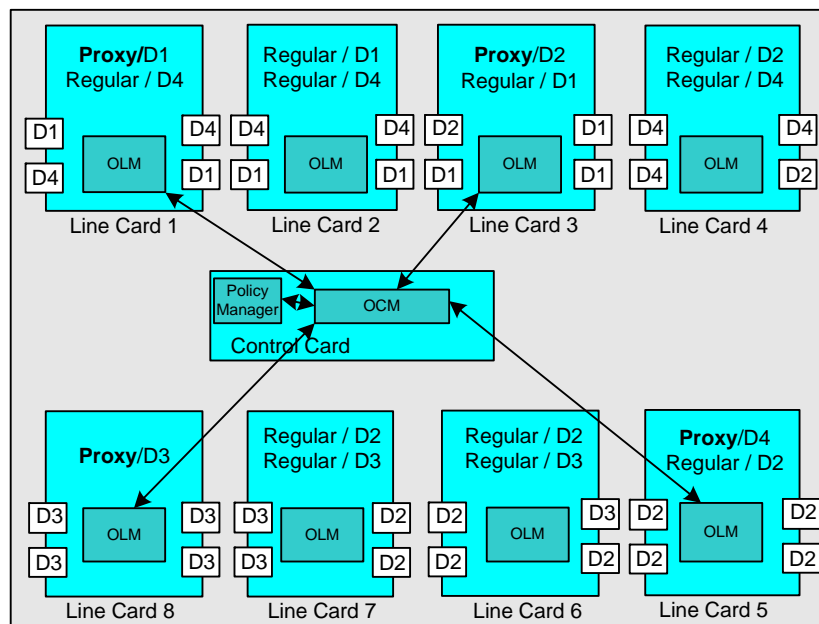


Figure 10: Management of OLM by groups

The proposed architecture also enables the direct communication among OLMs belonging to the same OSPF area, which is a great improvement compared to other OSPF architectures [14]. This allows each flooding message, generated by the router, to be sent only once instead of by all interfaces, hence it reduces the traffic in the network. Indeed, the OCM manages the OLMs by groups, and it uses a multicast address for internal communication among the members of each group. All OLM belonging to a group has at least one interface

connected to the same OSPF area, as shown in Figure 10. This management mechanism allows any member of a group to send messages to other members through the group multicast address. Each group has a “proxy” assigned by the OLM. When an OLM receives an LSA from a given OSPF area, it first checks its local LSA database and synchronizes with the proxy of the group corresponding to the area. The proxy is responsible to forward new or updated LSAs to the OCM, after being added into the local LSA database, in order to update the global LSDB. This scalable management mechanism reduces significantly the number of messages going to the control card.

Each OLM also sends a notification message to the OCM whenever there is a run-time error.

The adjacency management is also achieved at the line card level in the proposed architecture, in order to reduce the load on the OCM. Each OLM can discover the neighbors by itself because the Hello protocol is implemented on the line card.

### C. RSVP-TE/MPLS

The proposed distributed MPLS architecture using RSVP-TE as signaling protocol, aiming at improving the scalability of the current centralized architecture, comprises the following components (Figure 11) [16]:

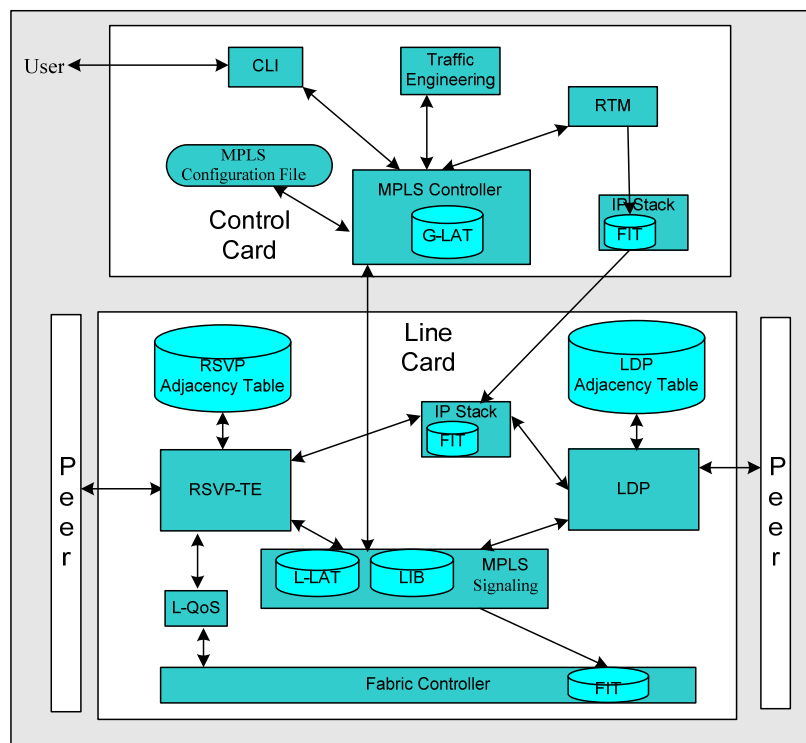


Figure 11: Distributed MPLS/RSVP-TE Architecture

- *MPLS data plane.* Although in the current centralized architecture, the data plane is implemented on the line cards, the processing task is carried out mostly at the ingress line card. We ensure a load balancing between ingress and egress line cards in the distributed architecture.

- *LAT (Label Allocation Table)*. Access to the LAT can be done at the line card level instead of the control card one. Processing speed is then accelerated as each line card can locally decide to generate labels according to the requests from its peers.
- *LIB (Label Information Base) table*. The current LIB is located on the control card and contains the overall LSPs of the system. It is then copied to the FIT of each line card. This is an extra overload since some LSPs never go through the given line card (this is because each line card hosts only some hundred LSPs while there can be hundred of thousands LSPs in the whole system). Distributing the LIB over the line cards accelerates the protocol processing and hardware recording. In addition, the memory requirement of the line card is alleviated by including only the required LSPs in the LIB of the line card.
- *Signaling protocol*. In the distributed architecture, the control card is no longer involved in the signaling message processing and transmission. The scalability and resiliency can be improved as discussed above. The adjacency manager is also offloaded to the line cards along with RSVP-TE module. This optimizes the adjacency tables because the line cards will manage only the neighbors to which they effectively communicate, while the current MPLS Controller on the control card has to manage all adjacencies of the system.

Indeed, migrating some of the processing tasks from the control card to line cards reduces potential bottlenecks experienced on the control card when the number of requests increases, according to the number of line cards and routes the core router has to support. The most important feature is that the proposed model can take advantage of the additional resource available on line cards of the next generation routers. In addition, it has the following advantages:

- *Performance*: Parallel processing is available and some waiting queues can be avoided. Line cards can independently process the routes they are involved in, with no need to wait for the reply from the control card.
- *Scalability*: The router will be more scalable if some control tasks, particularly the signaling, can be processed by line cards. The control card will assume only the most complicated tasks, the tasks that need human interactions or the tasks used to synchronize the processes running on different line cards.
- *Resiliency*: If the control card is required to perform all control tasks, the router will be totally shutdown when the control card fails. One of the possible solutions is to have an additional control card, to be used as a backup for the primary control card. However, control cards are often costly. Having a backup mechanism at the line card level as described in [12] provides a better solution: It is faster to recover from line card failures; moreover, a line card is much cheaper than a control card.
- *Availability*: Since HELLO messages can be sent directly from the line cards, the time to recover from failures will be reduced. The resulting congestion at the control card level will not slow down the procedures on the line cards.

On the other hand, the distributed architecture can raise some additional management

overheads, as follows.

- *Message exchanges between the control card and line cards:* Although the heaviest processing tasks have been eliminated on the control card (i.e., route computation and message sending/receiving), the control card is still responsible for the cooperation among the line cards. Such an activity is supported by an internal protocol, thus additional message exchanges are required.
- *The software complexity is increased:* The router can be seen as a completely distributed system hence additional functions must be implemented, such as timing and inter-card synchronization. Line card software should also provide extra-functions, such as message processing, table management or inter-protocol communications.

Although trade-offs have to be made due to the migration of control functions from the control card to the line cards, we believe that the proposed distributed architecture is a promising candidate for dealing with next generation router issues, particularly with a large number of line cards.

### 7. Performance Evaluation of distributed control plane architecture

We conducted an estimation of the performance achieved by the proposed distributed architecture, in terms of required memory, CPU cycles used and the number of exchanged messages. OSPF performance is measured in a petabit router model, so called PBR1280, developed by HyperChip, while RSVP-TE/MPLS and BGP experiments are performed on a PC-based platform simulation powered by PBR1280's software at Concordia University. For each router configuration, and each routing protocol module, we compare the performance achieved by the proposed and centralized architectures in terms of CPU cycles and the number of exchanged messages for up to 128 line cards. We work under the assumption of 10 ports per line card, with an increasing number (between 16 and 128) of line cards per router. The connectivity of the network is also taken into account throughout the number of ports per domain and the number of LSPs per port.

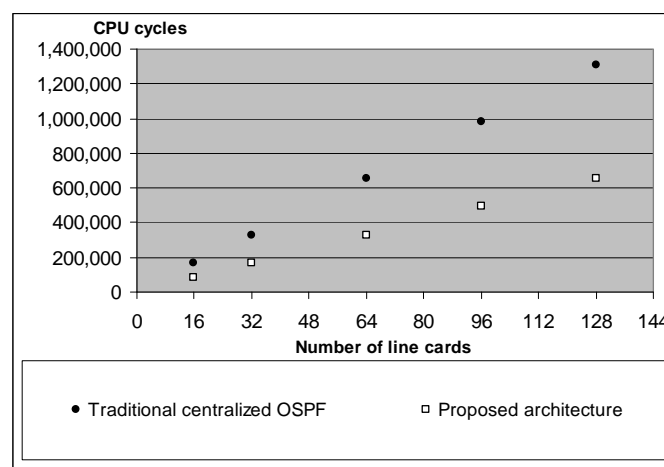


Figure 12: CPU cycles used for OSPF processing at control card in the centralized and proposed architecture

Figure 12 presents a comparison of performance achieved by the centralized and distributed OSPF architectures. The proposed distributed architecture saves considerable CPU resource on the control card as Hello protocol, message sending/receiving functions, as well as LSA message synchronization and validation are performed at the line card level. We can see that the CPU requirements for running OSPF at the control card is about two times less compared to the centralized architecture. The more line cards are added, the larger the difference is between these two numbers.

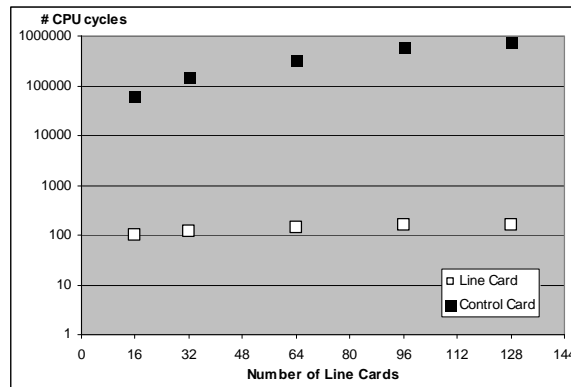


Figure 13: CPU resource consumption on control card and line card in the RSVP-TE/MPLS centralized architecture.

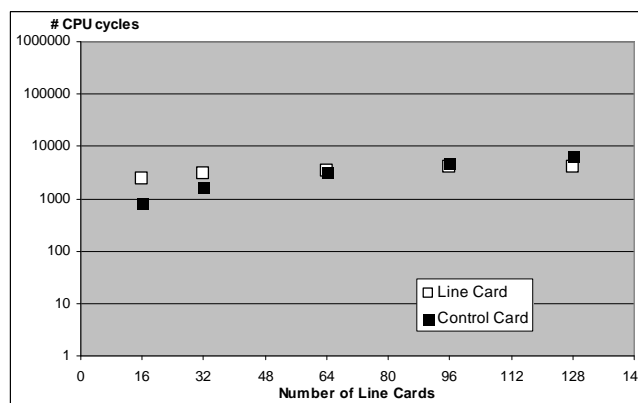


Figure 14: CPU resource consumption on control card and line card in the RSVP-TE/MPLS distributed architecture.

A similar comparison is performed for RSVP-TE, as described in Figures 13 and 14. As seen in the centralized architecture, the number of CPU cycles used on the control card is very high, i.e., 1,000 times more than in the distributed architecture for 128 line cards, when we increase the number of line cards and the number of LSPs per port. Consequently, in a centralized router, the number of line cards which can be served by a single control card remains limited. In the distributed architecture, the CPU load on the control card is lower because most of processing tasks are moved to the line cards such as message processing, label provisioning and table updates. This allows a control card to serve a larger number of line cards and RSVP-TE sessions. In the centralized architecture, the CPU utilization on line cards is low because it is required only for sending and receiving messages. In the distributed

architecture, the CPU requirement on line cards slightly increases in order to handle additional tasks (see Figure 14). However, it remains comparable in both architectures even for 128 line cards.

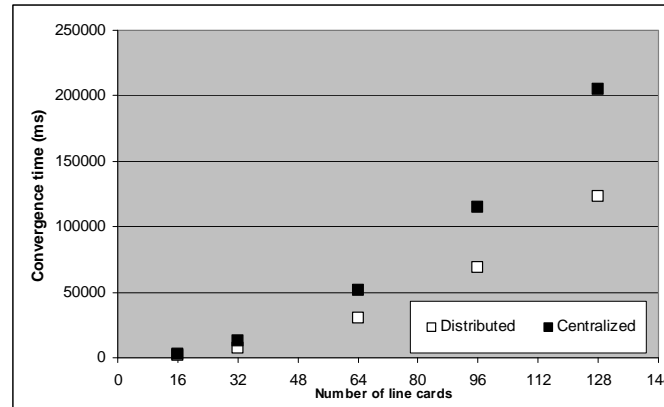


Figure 15: Average BGP convergence time in the BGP centralized architecture and a distributed architecture with 3 control cards.

We also compared the BGP centralized and distributed architectures in Figure 15, except that the convergence time is used as comparison criteria instead of CPU resource consumption as in the two previous cases, because it is considered as the most important characteristic of BGP performance. We can see that the convergence time for each BGP instance hosted by an independent control card in the proposed distributed architecture is much smaller than in the centralized architecture. This time can still be reduced if a more advanced distributed scheme is implemented, where some protocol processing tasks are performed at the line card level.

## 8. Open Issues

In order for the routers to become more scalable, one must shift towards a distributed control plane architecture to fully exploit the new router hardware architecture. However, a fully distributed architecture requires a large capacity of computing resources in the line cards and the switch fabric, which is not yet necessarily in the case of the current routers. Therefore, a transition period from monolithic to modular and then to distributed architecture can be a good solution. In such a period, control functions will be offloaded to the line cards over time, starting with the signaling functions, then route computation and finally the route management. Such an offloading seems easy for protocols such as OSPF, IS-IS, as well as LDP and RSVP. However, it remains a challenge for EGP protocols like BGP, due to its loop back addresses [16]. While a first distribution of BGP over a set of control cards can help and useful to improve the scalability, ultimately, it will have to take place on line cards in order to reduce the bottleneck issues at the control card level. When BGP is offloaded to the line cards, complex synchronization and message exchange mechanisms are required in order to resume BGP sessions over different line cards in case a session is interrupted. In addition, such a resume process needs to be transparent to users.



## 9. Conclusion

In this paper, we have discussed the evolution trends of the routing engine, the essential software component of current routers. The routing engine is composed of routing modules and the routing table manager, achieving the route establishment and forwarding table updates. We reviewed three main architectures for the routing engine, namely the monolithic, cluster-based and distributed architectures available in current router products. As the traffic in the core networks increases, we propose a new framework that is able to embed distributed architectures for routing protocols. Such a distributed architecture used for next generation routers *a priori* significantly enhances the FORCES architecture both in terms of scalability and resiliency. Scalability evaluation has been done in order to compare the proposed distributed architectures with centralized ones.

## References

- [1] Chao, HJ. (2002). Next Generation Router, *Proceeding of the IEEE*, vol. 90, no 9, 1518-1558.
- [2] Császár, A., Enyedi, G., Hidell, M., Rétvári, G., Sjödin, P. (2007). Converging the Evolution of Router Architectures and IP Networks, *IEEE Network Magazine*, vol. 21, iss. 4.
- [3] Handley, M., Greenhalgh, A. (2004). XORP: Breaking the Mould in Router Software, *Proceeding of London Communications Symposium*, UK.
- [4] Yang, L., Dantu, R., Anderson, T., Gopal, R. (2004). RFC 3746: Forwarding and Control Element Separation (ForCES) Framework, *IETF Network Working Group*.
- [5] Cisco Systems, (2000). Cisco 12000 Series Internet Router Architecture, [Online] Available: <http://www.cisco.com>.
- [6] Juniper Networks Inc., (2000). ERX router software overview, *White Paper*.
- [7] NextHop Technologies (2001). GateD releases (Web site). [Online] Available: <http://www.gated.org>.
- [8] Bollapradqda, V., Murphy, C., White, R. (2000). Inside Cisco IOS Software Architecture. *Cisco Press*.
- [9] Zebra project, (2007). GNU Zebra routing software. [Online] Available: <http://www.zebra.org>.
- [10] Nguyen, K.-K., Jaumard, B., Agarwal, A. (2008). A Distributed and Scalable Routing Table Manager for Next Generation IP Router, *IEEE Network Magazine*, vol. 22, iss. 2.
- [11] Nguyen, K.-K., Jaumard, B. (2007). A Distributed Model for Next Generation Router Software, *Proceeding of HPSR'07*.

- [12] Nguyen, K.-K., Mahkoun, H., Jaumard, B., Assi, C., Lanoue, M. (2007). Towards a Distributed Control Plane Architecture for Next Generation Routers, *Proceeding of ECUMN'2007*.
- [13] Decasper, D., Dittia, Z., Parulkar, G., Plattner, B. (2000). Router Plugins: A Software Architecture for Next-Generation Routers, *IEEE/ACM Transactions on Networking*, vol. 8, iss 1, pp. 2-15.
- [14] Deval, M., Khosravi, H., Muralidhar, R., Ahmed, S., Bakshi, S., Yavatkar, R. (2003). Distributed Control Plane Architecture for Network Elements, *Intel Technology Journal*, vol. 7, iss. 4.
- [15] Nguyen, K.-K., Jaumard, B. (2008). Distributed and Scalable Control Plane for Next Generation Routers: A Case Study of OSPF, *Proceeding of LCN'08*, Montreal, Canada.
- [16] Nguyen, K.-K., Jaumard, B., (2009). A Scalable and Distributed Architecture for BGP in Next Generation Routers, *Proceeding of ICC workshop 2009*.
- [17] Nguyen, K.-K., Jaumard, B., (2009). A Distributed and Scalable RSVP-TE Architecture for Next Generation IP Routers, *Proceeding of HPSR'09*.