Separating Control Software from Routers

R. Ramjee F. Ansari M. Havemann T.V. Lakshman T. Nandagopal K. Sabnani T. Woo
Bell Laboratories, Lucent Technologies
{ramjee, ansari, havemann, lakshman, thyaga, kks, woo}@lucent.com

Abstract— **Control software in routers have gotten increasingly complex today. Further, since the control software runs in every router, managing a large network of routers is complex and expensive. In this paper, we propose that the control software be hosted in a few control element servers remotely from the forwarding elements (routers). This reduces the software complexity in numerous forwarding elements, thus increasing the overall reliability of the network. In order to achieve this, we describe the design and implementation of two protocols: 1) Dyna-BIND that allows the forwarding elements to dynamically bind to control elements and 2) ForCES that allows the control elements to control the forwarding elements. Furthermore, we argue through several examples that the separation and logical centralization of control plane software in this architecture enables easier deployment of new services.**

I. Introduction

Network operators worldwide are currently contemplating a move towards a converged IP network in which they expect to carry voice, video, and data traffic. For example, British Telecom is launching a major initiative to move towards a converged 21st century IP network by 2010. IP routers comprise the basic network element in these converged IP networks. Thus, a closer examination of the architecture and functions of IP routers and networks is critical. Re-examining the distribution of router functions has been a topic of much recent research interest [2], [3], [18] (see Section VIII on related work for a detailed discussion).

In this paper, we focus on one critical aspect of IP routers: the control software executing on each of the routers. Despite the end-to-end architecture design principle that aims at a simple core network, routers have gotten increasingly complex today. As new features are being defined and standardized in RFCs, more and more control plane complexity is being added at the routers. These features include routing (e.g., BGP-based MPLS-VPNs), traffic engineering (e.g., OSPF-TE), security, etc. In fact, the code complexity of an IP router now rivals that of a 5ESS telephony switch[1]. In contrast, the forwarding path implementation has progressively become easier with rapid advances in large-scale hardware integration (e.g., ASIC) and ready availability of off-the-shelf chips.

[1]Approximately 5-10 million lines of code.

To make matters worse, the extremely complex control software executes on every router in the network. Given that a typical operator's network consists of hundred or more routers, managing the router control software (e.g. configuration, upgrades, maintenance etc.) is very expensive. Thus, this results in very high operational expenses for network operators[2].

The crux of the complexity issue in current routers is because implementations of the control and forwarding functions are intertwined deeply in many ways. The control processors implementing control plane functions are colocated with the line cards that implement forwarding functions and often share the same router backplane. Communication between the control processors and the forwarding line cards is not based on any standards-based mechanism, making it impossible to interchange control processors and forwarding elements from different suppliers. This also leads to a static binding between forwarding elements and line cards. A router typically has at most two controllers (live and stand-by) running control plane software. The two controllers, the line-cards to which they are statically bound, and the switch fabric together constitute the router.

In this paper, we argue that separating the software from the routers can significantly reduce the complexity in routers. To this end, we describe a control plane architecture called the SoftRouter architecture that separates the implementation of control plane functions from packet forwarding functions. In this architecture, all control plane functions are implemented on general purpose servers called the control elements (CEs) that could be multiple hops away from the line cards or forwarding elements (FEs). Each FE, when it boots up, discovers a set of CEs that can control it. The FE dynamically binds itself to a "best" CE from the discovered set of CEs. We envisage a standardized interface between the CEs and the FEs similar to that being standardized in the IETF ForCES working group [18].

One of the key benefits of separation of the control

[2]Total operational expenses may be anywhere between two to five times the capital expenses.

software from the numerous routers (forwarding elements) into a few centralized servers (control elements) is increased reliability. Given that software failures and configuration errors are the most common causes of failures, the dramatic reduction of software in majority of the elements in the network (the FEs) increases the reliability of the network significantly (see Section II). We argue that the forwarding element, apart from an IP protocol stack, needs only two key protocols: Dynamic binding protocol called Dyna-BIND that maintains the association between FEs and CEs and a FE-CE protocol called the ForCES protocol that allows the CEs to control the FEs. This dramatically reduces the amount of software in each FE while allowing the FEs to serve as fully functional routers that are controlled by the remote control elements. Another key advantage of moving the software away from routers is the ability to introduce new services easily and efficiently as discussed in Section VII. Apart from these advantages, this architecture also has higher scalability, lower cost, and increased security as argued in [12].

The rest of the paper is organized as follows. We provide motivation for separating software from routers in Section II. We present an overview of the SoftRouter architecture in Section III. We then discuss the design and implementation of the dynamic binding protocol that facilitates dynamic binding between FEs and CEs in Section IV. In Section V, we discuss design and implementation of the Forces protocol that is used for communication between the FEs and the CEs. In Section VI, we present our testbed. In Section VII, we illustrate how it is easier to deploy new services in this architecture. We present related ork in Section VIII. We finally present our conclusions in Section IX.

II. Motivation

The desire to split the control and forwarding planes in routers arises from multiple factors. These factors arise out of various technological, engineering and economic issues. Apart from the factors identified in Section I, the improved reliability of the network is a key enabler for the proposed separation.

The SoftRouter architecture has multiple forwarding elements controlled by a single control server. The forwarding elements are primarily hardware-based systems with a control element of very minimal functionality and an elementary software component. The control server participates in the routing protocols on behalf of the forwarding elements (FEs) and computes the routes for all the FEs under its control. The software complexity resides principally at the control servers (CEs). There are N FEs and K CEs to control them, in the network, where $K \leq N$.

A. Component Reliability

We first consider the benefits of choosing the SoftRouter approach over the traditional distributed router approach, by looking at the probability of network disruption due to failure of individual components in the network.

We will first assume that the only parts that can fail in the network are the *line blades, control cards, and operating software*. In a distributed router architecture, let the probability of failure of each unit of line card, control card and operating software be p_L, p_C, and p_S, respectively. We will relax this assumption later to include more failure cases, including link failures. Let the total number of line cards in a router (and an FE) be M.

In the SoftRouter architecture, each FE has multiple line cards, a low- capacity control card, and simple software. Let the failure probability of each of these units be p_L, p'_C, and p'_S, respectively. At the CE, there is one control card, a line card, and the bulk of the routing software, with failure probabilities of p_L, p_C, and p_S, respectively. Note that $p'_C << p_C$, and $p'_S << p_S$, since the control components (namely the control card and the software) are more than an order of magnitude complex at the CE than at the FE.

Ideally, we do want anything in the network to fail at all. Thus, we first calculate the probability that no component fails in the network. In the distributed model, this turns out to be $p^{Dist}_{NoFail} = [(1-p_L)^M(1-p_C)(1-P_S)]^N$. For the SoftRouter model, the probability that no component fails in the network is given by $p^{SoftR}_{NoFail} = [(1-p_L)^M(1-p'_C)(1-P'_S)]^N[(1-p_L)(1-p_C)(1-P_S)]^K$.

Given that $p'_C << p_C$, and $p'_S << p_S$, the *no-failure probability* is *always much higher for the SoftRouter than for the traditional fully distributed router architecture*, as can be seen from the above equations, except when K is close to N. The difference becomes more pronounced as the number of FEs (or routers, in the traditional sense) increase. Given that 25%-33% of network outages are caused by software failure in the control card, this improvement in no-failure probability in the SoftRouter is highly desirable.

Figure 1 compares the no-failure probability of the network to various software failure probabilities, and evaluates both traditional and SoftRouter architectures in this model. The line card and control card failure probabilities are 0.00001 and 0.0001 respectively. It can be seen that for current software reliability estimates (between 99.9% and 99.99%), the SoftRouter architecture performs much better than the traditional model, with greater improvements as software reliability decreases.

Figure 2 plots the failure probability of the control element (CE) against the network no-failure probability. The results are interesting: SoftRouter boxes can afford to be

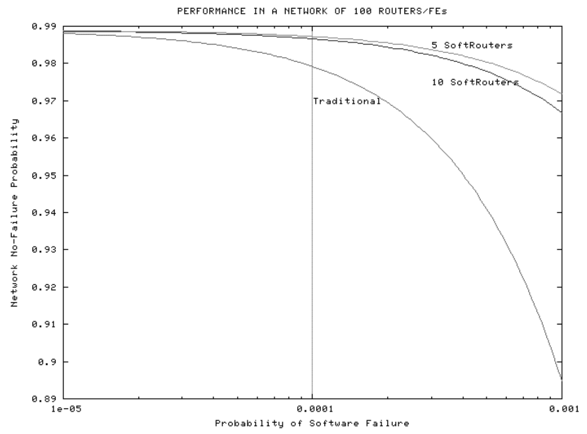

Fig. 1. Impact of software failure probability

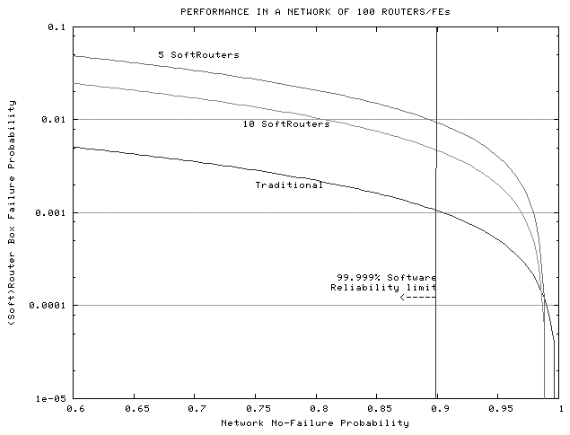

Fig. 2. Impact of control element failure

less reliable individually than a traditional router, by up to an order of magnitude, without compromising the network no-failure probability. The only time when a traditional router offers better performance is when the software is close to completely reliable (better than 99.999%), which requires that the software is more reliable than the underlying hardware. Thus, the SoftRouter architecture relaxes reliability requirements on the control element (CE), while providing the same levels of performance as traditional routers.

B. Link Failures and Reliability

We will now include link failures in the no-failure model discussed above, and analyze the SoftRouter architecture, since earlier studies have shown that nearly 25%-40% of network outages are caused by link failures.

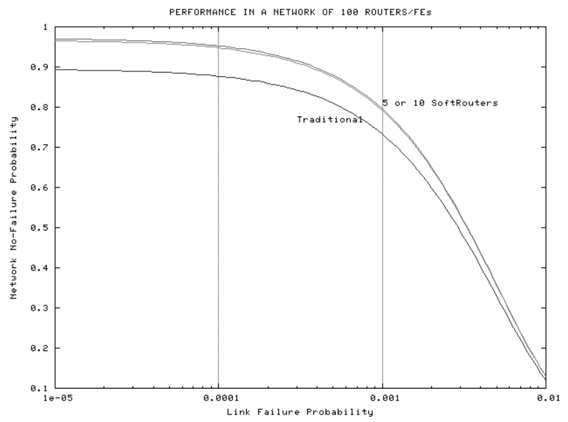

Fig. 3. Impact of link failure probability

In the SoftRouter architecture, the number of links in the network is slightly higher than the number of links in a traditional router network. The additional links are the ones that connect the CEs to the network. Given a port density of D per line card, we can assume that there are at most D links in each SoftRouter, with a total of $K.D$ additional links in the network. Define the failure probability of a link to be p_E. Let the total number of links in a traditional network be E. Let us assume that the number of links connecting the SoftRouter CEs to the network, $K.D$, is smaller than N, i.e. $K << N$. Therefore, the above no-failure equations can be modified as follows.

$$p^{Dist}_{NoFail} = [(1-p_L)^M(1-p_C)(1-P_S)]^N(1-p_E)^E$$

$$\begin{aligned} p^{SoftR}_{NoFail} &= [(1-p_L)^M(1-p'_C)(1-P'_S)]^N \\ &\quad [(1-p_L)(1-p_C)(1-P_S)]^K(1-p_E)^{E+KD} \end{aligned}$$

Thus, the improvement in the no-failure probabilities of the SoftRouter architecture vis-a-vis the fully distributed router architecture is reduced by only a small factor of $(1-p_E)^{KD}$ when compared to the case that ignores link failure probabilities.

In Figure 3, we plot the network no-failure probability against various values of link failure probabilities in the network. We assume a network of 100 nodes with 400 links. We assume that each SoftRouter has two links connecting it to the network (D=2). We fix the various hardware failure probabilities as in the previous figures, and assume that the software is 99.9% reliable. The results shown below indicate that when links fail frequently, they tend to be the biggest factor that influence network no-failure probability. However, the SoftRouter model consistently has better no- failure probability than a traditional router model, regardless of the link failure probabilities. When the software is 99.99% reliable, the difference becomes smaller but is still in favor of the SoftRouter model. Only when the software is 99.999% reliable does the performance of the two architectures become equal. Taking this fact in consideration along with the lower reliability requirements of the SoftRouter, we have a clear case in favor of employing the SoftRouter architecture in current networks.

III. Architecture

While the focus of this paper is on the software protocols that are essential in the forwarding elements, it is necessary to understand these protocols in the context of the overall architecture. Thus, in this section, we present an overview of the SoftRouter architecture that was originally introduced in [12].

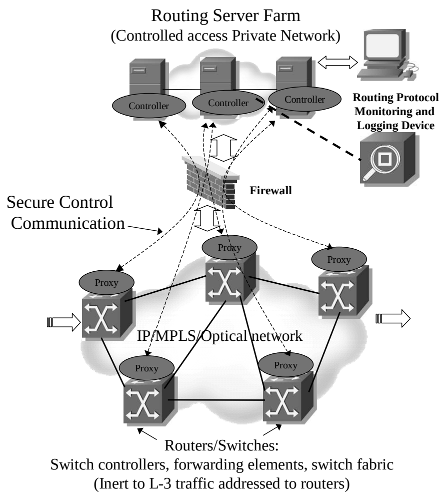
Routing Server Farm
(Controlled access Private Network)
Controller
Controller
Controller
Routing Protocol Monitoring and Logging Device
Firewall
Secure Control Communication
Proxy
Proxy
Proxy
Proxy
Proxy
IP/MPLS/Optical network
Routers/Switches:
Switch controllers, forwarding elements, switch fabric
(Inert to L-3 traffic addressed to routers)

Fig. 4. Logically separate control and data planes

A. Network Entities

As mentioned earlier, there are two main types of network entities in the SoftRouter architecture, the FEs and the CEs, that together constitute an NE (router).

Forwarding Element (FE): FE is a network element that performs the actual forwarding and switching of traffic. In construction, an FE is very similar to a traditional router; it may have multiple line cards, each in turn terminating multiple ports, and a switch fabric for shuttling data traffic from one line card to another. The key difference from a traditional router is the absence of any sophisticated control logic (e.g., a routing process like OSPF or BGP) running locally. Instead, the control logic is hosted remotely. The exact nature of forwarding function can be (1) Packet forwarding: this includes both layer 2 (MAC-based switching) and layer 3 (longest-prefix match) forwarding. (2) Label switching: an example of this is MPLS forwarding. (3) Optical switching: the traffic in this case can be time-switched, wavelength-switched, or space-switched among the links. In each of these cases, the switching function is driven by a simple local table which is “computed” and “installed” by a CE in the network.

Control Element (CE): A CE is essentially a general purpose computing element, such as a server. It connects to the network like an end host, except that it is typically multi-homed to the network via multiple FEs, so that it is not disconnected from the network when a single link fails. A CE runs the control logic on behalf of FEs, and hence “controls” them. In principle, any control logic typically found on a traditional router can be migrated to the CEs; these include routing protocols like OSPF and BGP as well as protocols such as RSVP, LDP, Mobile IP, etc.

Network Element (NE): At a high level, an NE is a logical grouping of FEs and the respective CEs that control those FEs. Given this wide spectrum of possibilities of FE/CE combinations, we focus on a restricted but practical case where the FEs making up an NE are part of a contiguous “cloud.” Physically, this represents the clustering of neighboring physical forwarding elements into a single NE. A typical scenario is that of several routers being connected back-to-back in a central office. From a routing perspective, this clustering-based definition of the NE results in a natural hierarchy, thus reducing the inter-NE routing complexity.

B. Network Architecture

There are two possible ways of separating the CEs (control plane) from the FEs (data plane). In a logical separation, as shown in Figure 4, a SoftRouter network is not significantly different from a traditional routed network, except for the addition of a few multi-homed servers (CEs). The control plane protocol messages continue to traverse the data plane for communication between adjacent routing peers. This results in an architecture that very closely resembles the current architecture except for the decoupling of the control plane and the resultant benefits of improved scalability and reliability. Since it mimics the current network architecture, minimal routing protocol changes are needed for proper functioning. In a physical separation (see Figure 5), the control plane is physically separated from the data plane, similar to the way the SS7 signaling network is separate from the telephony network. Thus, all controllers in the routing server farm form their own private network topology that is independent of the underlying forwarding plane topology. This provides for

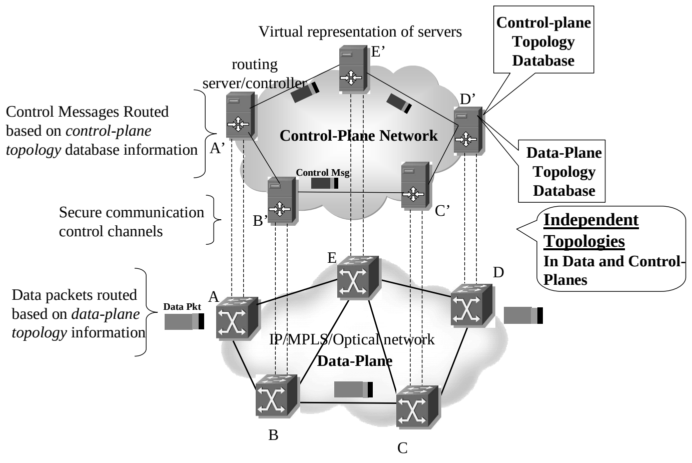
Virtual representation of servers
Control-plane Topology Database
routing server/controller
E'
D'
Control Messages Routed based on *control-plane topology* database information
A'
Control-Plane Network
Data-Plane Topology Database
Control Msg
Secure communication control channels
B'
C'
Independent Topologies In Data and Control-Planes
E
D
Data packets routed based on *data-plane topology* information
A
Data Pkt
IP/MPLS/Optical network
Data-Plane
B
C

Fig. 5. Physically separate control and data planes

a very high security environment for the network, in addition to the improved scalability and reliability advantages mentioned above. However, the downside of this architecture is that changes to existing routing protocols are needed - e.g. the protocols need to keep track of two network topologies, one for the data network and other for the control network.

C. Protocols

The focus of this paper is on the building blocks, common to both of the above architectures, that allow the data plane to be decoupled from the control plane. Specifically, two protocols serve as these building blocks. First, each FE must discover and bind with its controlling CE - this is achieved using the Dyna-BIND protocol described in Section IV. Second, once an FE is bound to its controlling CE, we need that a protocol that will help in exchanging state (link up/down etc.) and allowing CE to perform control over the FE (e.g. enable/disable a link, change forwarding information base, etc.). This function is achieved using a standards-based ForCES protocol [18] that is described in Section V.

IV. Dynamic Binding Protocol

The Dynamic Binding Protocol (Dyna-BIND) provides binding service for FEs and routing services for FE-CE messages. Dyna-BIND runs continuously on all the elements (FEs and CEs) for the lifetime of the network as a separate protocol, independent of other protocols that operate in the network. We assume that at the minimum, each FE and CE has a pre-configured octet string (FEID/CEID) that acts as a unique identifier. In a bridged network of FEs and CEs (for e.g., connected over Ethernet), a rapid spanning tree protocol can provide the connectivity for FE-CE communication. Ethernet networks are only a small fraction of possible networks. Dyna-BIND includes a separate component for supporting protocol independent routing services between the CEs and FEs. Dyna-BIND may run on layer2 or layer3.

Dyna-BIND has four components:

- Discovery
- Association
- Failure Detection with Repair
- Transport Tunnels.

We now discuss each of these components. For simplicity, we will assume that the entire network consisting of CEs and FEs forms a continuous domain. Additionally we assume that Dyna-BIND is running over IP.

A. Discovery

The discovery component in Dyna-BIND enables an FE to discover a CE, which can manage it. This CE is called the managing-CE for that FE. It is highly desirable for the discovery component to provide rapid convergence for the configuration process. The goal is to distribute CE information all over the network, thereby enabling FEs to dynamically bind to the best CE using bindings either pre-configured by the network administrator or obtained using distributed clustering algorithms.

We define a source-routed routing layer to help in the discovery process. At the time of boot-up, each FE uses a set of randomly chosen temporary IP addresses for its interfaces, along with the FEID, in order to perform pre-discovery routing. This address is chosen from a controlled address space; for example, it could be a private subnet address (such as 10.*.*.*). The addresses have to be unique only on a per-link and per-node basis. The CEs are pre-configured and hence have valid IP addresses. FEs and CEs discover their neighbors by advertising their presence to their immediate neighbors by periodically multicasting HELLO messages. Each node (FE/CE) thus will maintain a list of neighbors in a local table, which maps FEIDs/CEIDs to IP addresses and interfaces.

A source route is a recorded sequence of FEIDs and CEID, and is part of the Dyna-BIND packet header. At each hop, the next hop FEID or CEID is translated into a next hop IP address taken from the neighbor table. This IP address can be a unicast or multicast IP address. Whenever possible messages are sent as unicast messages to limit the performance impact of multicast messages. Only HELLO messages are sent as multicast at all times. If a neighboring node shares the same IP network, messages are sent as unicast messages. If the neighboring node belongs to a different IP network, messages are sent as multicast. A node determines the IP network by comparing the source address of a received HELLO message with its interface address. (Note that the scenario of neighboring nodes belonging to different IP networks can occur when an unconfigured FE peers with configured FEs or CEs.)

CEs flood their identity periodically and in response to network events throughout the network with advertisement messages. The identities are propagated reliably on a hop-by-hop basis. FEs maintain a CE-reachability table that records CEIDs along with a source route to the CE, a sequence number and a time-to-live value. When an FE receives an advertisement it updates the corresponding entry in its reachability table. An entry is updated if the sequence number of the message is newer than the one in the table. The advertisement is then transmitted on all interfaces except the one it was received on. If the sequence number is older, then the message is ignored and not propagated further. If the sequence number is the same, the hop count of the advertised source route will be analyzed. If the new source route is shorter, then the table will be updated and the message passed on. If the source route has the same length, then the message will be ignored, thereby bounding the flood.

Sequence numbers are reset with a special sequence number reset message when the linear sequence space is used up or when a CE reboots.

If an FE detects that a link to a neighbor is down, it then sends an acknowledged link-failure notification message to those CEs that are upstream from the FE with respect to the broken link. On reception of a link failure notification, a CE will flood its identity immediately to re-establish source routes. Floods in response to link failure notifications are rate-limited in order to prevent network overload. Advertisement messages are flooded individually and are not aggregated. The underlying assumption is that a SoftRouter network has only very few CEs compared to a large number of FEs. When a link is restored, the corresponding upstream FE sends an acknowledged link restoration notification to the CEs in its reachability table, thus triggering an advertisement flood.

CEs also maintain a reachability table. FEs send periodic heartbeat messages (described below) besides other Dyna-BIND messages. A CE extracts the source route from these messages, reverses it and then stores it along with a time-to-live value in its reachability table.

B. Association

Each FE is assigned one primary-CE and at least one backup CE by the network administrator during network planning. This information is configured in the CEs, and optionally, in the FEs. Typically, this assignment is made a-priori by taking into account factors such as the load on the CE, the distance between the CE and the FE, and the reliability of the links between the CE and the FE. Thus, when a CE is contacted by an FE, it lets the FE know the identity of its primary and backup CEs, if this information is available, or accepts the FE if it can manage it. If not accepted, the FE then proceeds to find and contact its primary/backup CEs. The association process strives to establish and maintain an active association between an FE and its primary-CE and backup-CEs. The CE currently controlling an FE is the managing-CE and is chosen by the FE from the list of actively associated CEs, with the primary-CE preferred at all times to other CEs.

C. Failure Detection and Repair

The Dyna-BIND protocol has mechanisms to detect and repair association failures. Once an association is made between the FE and a CE, the liveness of the association is probed periodically through heartbeat messages initiated by the FE. When heartbeat messages do not elicit any responses from the CE, it implies that either the path to the CE is no longer valid, or that the CE node is no longer alive.

Link failures in the immediate neighborhood are detected via exchange of HELLO messages between neighbors. If alternate paths are available, then the Dyna-BIND protocol uses them to probe association-liveness of the CE. If the CE in question was the managing-CE for the FE, and no alternate path is found, then the FE activates another associated CE from its associated CE list to become its managing-CE. It is critical for this failover to occur as soon as possible with minimal delay. Hence, it is imperative to decide which CEs are designated as backup CEs for a given FE. Note that an FE can have more than one backup CE with an order of preference among them. In the SoftRouter architecture, the criteria for selecting backup-CEs are that (a) the failover time be minimal, which is achieved by means of active associations in the Dyna-BIND protocol, and (b) there exists a path to reach a backup-CE inspite of link failures. The backup-CEs are chosen such that each backup CE has the least amount of path overlap with the previous backup-CEs and the primary CE. For example, the second backup-CE is chosen such that its path to the FE has the least overlap with the shortest paths from the FE to both the primary-CE and the first backup-CE. The goal is thus to ensure that connectivity is maintained even in the presence of multiple link or CE/FE failures. A FE that is using a backup-CE will always try to re-associate with its primary-CE and switch to it when conditions permit. This is to ensure that the load on all CEs remains equal, as decided by a load-balancing algorithm.

D. Transport Tunnels

The last component of the Dyna-BIND protocol sets up a rudimentary transport tunnel between an FE and its associated CEs, using the slow-path (i.e., the source-routed layer provided by the discovery component) and allows for CE-FE communication when all other communication means fail. These tunnels are unreliable and do not guarantee in-sequence delivery. They only provide some path between CE and FE but which is not necessarily the best path. It is important to understand that these sub-optimal tunnels are used only rarely, when there are no valid routing tables installed on the FEs, for e.g., when the FE is being initialized or when the FE is switching to a different managing-CE.

In summary, the combination of these four components helps the Dyna-BIND protocol to actively discover and maintain dynamic bindings between FEs and CEs in the network. We next discuss the ForCES protocol and then describe our implementation of these protocols in a testbed.

V. ForCES Overview

The ForCES working group at the IETF is chartered to define a framework and associated mechanisms for standardizing the protocol information exchange between the control plane and the forwarding plane. Some of the main results of this working group is to produce:

- A set of requirements for mechanisms to logically separate the control and data planes of the IP network element
- An architectural framework defining the entities comprising a ForCES network element and identifying the interactions between them
- A description of the functional model of a forwarding element and the formal definition of the controlled objects in the model.
- Specification of a set of communication protocols within the framework architecture. It includes the *ForCES protocol* to communicate between the CE and FE and an IP-based transport protocol that will be used to carry the messages between the elements.

In this section we will primarily focus on the ForCES protocol.

A. ForCES Protocol

The ForCES protocol works across the Fp reference point in the ForCES framework architecture [18]. Since the SoftRouter architecture is a realization of the ForCES framework, we use the ForCES protocol exactly for communication between the SoftRouter control and forwarding elements. Figure 6 provides an example of various

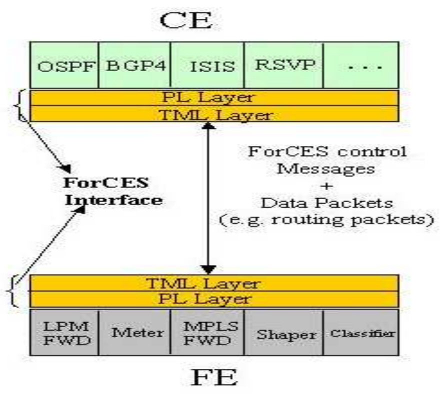

Fig. 6. ForCES Protocol and example modules on CE and FE

modules executing on the control and forwarding elements and using the ForCES protocol for communication between them. Note that control plane packets arriving from other nodes in the network are depicted as ”data packets” within the ForCES protocol - since these packets represent the payload, while the protocol has it’s own control messages purely operating between the CE and the FE. The ForCES protocol operates in a master-slave mode in which the FEs are slaves and CEs are masters.

The ForCES interface is broken down into two parts: the Protocol Layer (PL) and the Transport Mapping Layer (TML). The PL layer is in fact the ForCES protocol that defines all the semantics and the message formats, while the TML layer is used to connect two ForCES PL layers on the CE and FE respectively.

1) The PL Layer: The PL layer [1] is responsible for the setup and teardown of association between the FEs and CEs of an NE. It defines the ForCES protocol messages, the protocol state transfer scheme, as well as the ForCES protocol architecture itself. A CE uses the PL layer to activate, de-activate, subscribe to specific events, configure etc. an FE. An FE uses the PL layer to provide information on various status requests issued from a CE or generate event notifications based on subscribed-to events by the CE. A number of messages are defined at the PL layer for protocol operation. The PL delivers the messages to the TML layer, which in-turn delivers it to the destination TML/PL layer. The messages defined are given below:

- Association setup message
- Association setup response message
- Association teardown message
- Config message
- Config response message
- Query message
- Query response message
- Event notification message
- Event notification response message

- Packet redirect message
- Heartbeat message

2) The TML Layer: The TML layer [11] is responsible for transporting the PL layer protocol messages. It is in the responsbility of the TML layer to handle issues such as message reliability, ordering, congestion control, multicast etc. In the SoftRouter architecture we adopted the use of a TCP/IP based TML layer since most of the transport issues mentioned above are handled well by TCP - except multicast. Multicasting can be achieved by setting up multiple TCP connections between the CE and the FEs. Using TCP also enables the TML and the protocol to work seamlessly in single-hop and multi-hop environments.

VI. Implementation

Dyna-BIND runs on the CEs and FEs as a single task around an event driven state machine. A main event loop is triggered by message reception and timer events. Dyna-BIND messages are exchanged over UDP/IP over a well-known port. The CE structure is a bit more complex since a task may host multiple CEs, which share the well-known port. The receive function has to partially parse the packets to determine the CEID and lookup the corresponding CE context. On the FE side it is assumed that there is only a single logical FE instance per FE (i.e. one FE task per chassis blade or pizza box). Although a single task approach provides the highest performance, it comes with the cost of less reliability. A crash in one of the CEs will inherently bring down all other CEs. However, in general it is believed that performance is the more important property and stability can be achieved through thorough testing and capabilities for a fast restart.

We encountered an unexpected problem due to the source route filtering functionality present in today's operating systems such as Linux. This filter drops all received packets with a source address and receive interface pair that is not recognized by the routing table. The intention is to protect against denial of service attacks that uses spoofed source addresses. In the SoftRouter architecture, it is a common case during bootup (the FE configuration phase) that neighboring FEs belong to different IP networks for a transitional period until their configuration information is obtained from the CEs. Disabling source route filtering solved this problem.

Another challenge was to determine the right heartbeat timeout value for a CE fail-over. If we are too aggressive with the timeout value, due to variations in the flooding of CE advertisements, some FEs in the test network started to fail-over to their backup CEs before the source routes converged. On the other hand, a high enough timeout value of several seconds solves this problem but results in slower failure detection and recovery. A timeout value of several hundred milliseconds is a reasonable compromise. However, note that this timeout values is dependent on the time to flood the complete network, which in turn depends on the network span.

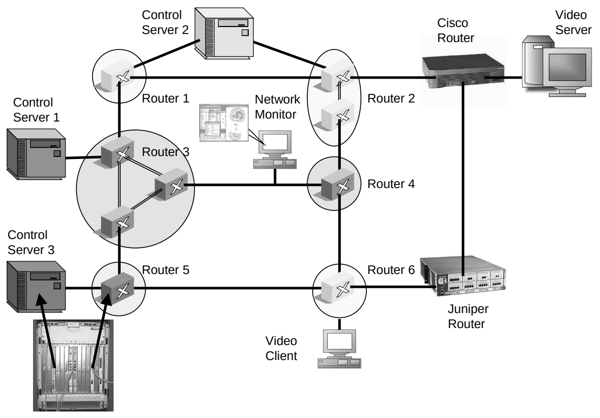

Fig. 7. Testbed implementation

We implemented the two protocols, Dyna-BIND and ForCES, in a testbed of FEs and CEs as shown in Figure 7. The testbed consists of 6 NEs labelled Routers 1 through 6 (each NE consisting of 1, 2, or 3 FEs) managed by three CEs labelled control Server 1 through 3, one Cisco 7200 router, one Juniper M-20 router, a network monitor, and one video server that is continously transmitting video to a video client.

In current routers with at most 1:1 redundancy of controller cards, if both controller cards are down, the entire router is declared down even though all the line-cards may be functioning properly. One of the advantages of decoupling the data plane and control plane in the SoftRouter architecture is the unique ability of performing a network-wide control plane failover, i.e., a totally different CE can takeover the control plane functions of an FE when the FEs primary CE fails. In this testbed, we were able to demonstrate a network-wide control plane failover: when the node labelled Control Server 1 is brought down, the control functions of node labelled Router 3 (such as OSPF protocol processing etc.) are taken over by the node Control Server 2, thus achieving higher network reliability.

VII. Services

In this section, we illustrate the rich service support that is possible in the SoftRouter architecture. We discuss the support for the following three services: a) overlay network support, b) Mobile IP support, and c) Virtual Private Network support. A common theme in enabling these new services is the centralized nature of control in the SoftRouter architecture.

A. Overlay network support

Overlays architectures and testbeds are becoming commonplace in the Internet today. For example, PlanetLab [13] provides an open testbed with planet-wide scalability including over 200 nodes distributed in over 60 countries in the world. There has even been recent discussion in the research community as to whether overlay should be part of a core IP service in next generation Internet [14].

Today, however there is no coordination between IP routing and overlay routing as these operate independently in different protocol layers. This lack of coordination can result in several issues [15] such as load balancing and traffic engineering problems for the ISP and network stability problems due to different interactions between overlay and IP routing.

A SoftRouter network can provide a hosted overlay IP service without resulting in any of the aforementioned drawbacks by having the CE perform coordinated IP and overlay routing. Recall that CE performs centralized routing in the SoftRouter architecture and can easily take the current overlay demand into account while performing its routing computations. Thus, the SoftRouter architecture, by consolidating the control functions in a few places in the network, is better positioned to deliver overlay as a core IP service than today's distributed Internet router architecture.

B. Mobile IP support

We now highlight the scalability advantage of the SoftRouter architecture in achieving a highly scalable Mobile IP home agent [16] service. Mobile IP home agent service will require increasing scalability as cellular carriers such as Verizon and Sprint introduce wireless data. Currently Nextel (iDEN) and SK Telecom (Korea/cdma2000) support Mobile IP in their networks. Verizon, Sprint, and others are expected to enable Mobile IP in order to support ubiquitous wireless data as they introduce CDMA EV-DO networks nationwide. There are two approaches to home agent scalability in the industry today: one approach is the use of routers and another approach is the use of general purpose processors. However, both of these approaches have limitations as discussed below.

Routers from major router vendors support several hundred thousand home agents but signaling scalability is limited to about hundred bindings/sec due to limitations of the control processor. In other words, signaling scalability is limited to less than two updates per hour per user. This is a significant limitation as updates generated through both mobility as well as the standard refresh mechanism built into Mobile IP can easily exceed two per hour per user. Mobile IP could also be implemented on a cluster of general purpose processors. Signaling scalability will not be an issue here. However, scaling the number of home agents becomes difficult since IPSec processing (for each tunnel - one per agent) is CPU intensive and will not be able to scale efficiently to several hundred thousand home agents without specialized hardware.

SoftRouter architecture admits a complementary combination of both of these approaches. It allows server based signaling scalability while retaining hardware based transport scalability. Thus, transport will still handled by FEs with hardware support for IPsec using regular router blades while signaling capacity can easily be scaled using multiple server blades, enabling sixty updates per hour per user or more.

C. Virtual Private Network support

There has been significant recent activity in defining network-based VPN services using BGP/MPLS [17]. In this application, a VPN server dynamically creates MPLS or IPSEC tunnels among the provider edge routers. While the VPN server would execute on the router control board in today's architecture, migrating the VPN server functionality into a control element in the SoftRouter architecture has several benefits: 1) Configuring BGP policies for the provider edge routers connected to the VPN customer sites can be done in a central location at the VPN server rather than at multiple routers (e.g. edge routers and route reflectors involved in the VPN); 2) MPLS tunnels can be engineered in a centralized manner to meet customer requirements; 3) Scalability for support of large number of VPNs can be easily handled using generic server scaling techniques; 4) Network-wide failover of VPN control servers can be performed without impacting existing or new VPN sessions; 5) VPN server upgrades can now be independently performed without impacting basic network operations such as forwarding.

VIII. Related work

The proposed network evolution has similarities to the SoftSwitch based transformation of the voice network architecture that is currently taking place. The SoftSwitch architecture [5] was introduced to separate the voice transport path from the call control software. The SoftRouter architecture is aimed at providing an analogous migration in routed packet networks by separating the forwarding elements from the control elements. Similar to the SoftSwitch, the SoftRouter architecture reduces the complexity of adding new functionality into the network.

One of the key benefits of the SoftRouter architecture is that it makes it easier to add new functionality into the network as discussed in Section VII. Researchers have proposed other techniques such as Active Routers [8] or open programmable routers [9] to increase flexibility in deploying new protocols in the Internet. By separating the forwarding and control elements and hosting the control protocols on general purpose servers, more resources are available for adding new software services in the SoftRouter architecture.

The Open Signaling approach [6] advocated the separation of control plane from the forwarding plane in ATM networks for increased network programmability. This separation enabled the Tempest framework [7] to create architectures where multiple control planes could simultaneously control a single network of ATM switches.

The Internet Engineering Task Force (IETF) is working on standardizing a protocol between the control element and the forwarding element in the ForCES [18] working group. Although the current focus of the working group is limited to single-hop, direct connection between the control element and the forwarding element, the set of protocols developed has been enhanced and employed in the SoftRouter architecture for communication between CEs and FEs as discussed in Section V.

The case for separating some of the routing protocols (specifically, BGP) multiple hops away from the routers have been made by several researchers [2], [4]. While it is possible to migrate a few selected protocols out of the forwarding element, such an approach does not deliver the full benefits of the SoftRouter architecture where apart from the Dyna-BIND and ForCES protocols, all other protocols are moved into control elements.

Authors in [3] propose a similar separation of functionality from the routers. However, their motivation is somewhat different - they would like to support centralized management of packet networks without being encumbered by the current distributed control plane implementation. Thus, they suggest that the current control plane software running in the routers be made as “thin” as possible and most of the control plane functionality be moved into the management plane.

IX. Conclusions

In this paper we presented the SoftRouter architecture where the forwarding elements were simple hardware devices with minimal software and were controlled by control elements that may be located remotely. We then focused on the two basic protocols that are necessary to enable this architecture: 1) Dyna-BIND that allows the forwarding elements to dynamically bind to control elements and 2) ForCES that allows the control elements to control the forwarding elements. We implemented these two protocols and demonstrated network-wide failover in a testbed of forwarding and control elements. Furthermore, we argued through several examples that the separation and logical centralization of control plane software in the SoftRouter architecture enables easier deployment of new services. As data networks become integral to everyday life and as new services become increasingly deployed on data networks, reliability and new service deployment become critical necessities. SoftRouter architecture is well placed to meet these critical requirements.

References

[1] A. Doria, Ed., ”ForCES Protocol Specification,” Working Group Internet Draft, draft-ietf-forces-protocol-04.txt, 2005.

[2] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, ”The Case for Separating Routing from Routers,” *Proc. of FDNA workshop*, August 2004.

[3] J. Rexford et. al., “Network-Wide Decision Making: Toward A Wafer-Thin Control Plane,” *HotNets*, 2004.

[4] R. Govindan et. al., ”Route servers for inter-domain routing,” *Computer Networks and ISDN systems*,Vol. 30, 1998, pp 1157-1174.

[5] S. Williams, ”The softswitch advantage,” *IEE Review* Volume: 48 , Issue: 4 , July 2002, Pages:25 - 29

[6] A. Lazar, “Programming Telecommunication Networks,” *IEEE Network*, vol 11., pp. 8-18, Sept/Oct. 1997.

[7] Sean Rooney et. al., “The Tempest, a Framework for Safe, Resource Assured, Programmable Networks,” *IEEE Communications*, Vol 36, No. 10, Oct. 1998, pp.42-53.

[8] D. Wetherall, U. Legedza and J. Guttag, ”Introducing New Internet Services: Why and How,” IEEE Network Magazine, July/August 1998.

[9] M. Handley, O. Hudson, and E. Kohler, ”XORP: An open platform for network research,” *HotNets*, Oct. 2002.

[10] Ram Keralapura et. al., “Service Availability: A New Approach to Characterize IP Backbone Topologies,” IWQoS, 2004.

[11] H. Khosravi, F. Ansari, J. Maloy and S. Chawla, ”TCP/IP based TML (Transport Mapping Layer) for ForCES protocol,” Working Group Internet draft, draft-ietf-forces-tcptml-01.txt, 2005.

[12] T.V. Lakshman, T. Nandagopal, R. Ramjee, K. Sabnani, and T. Woo, “The SoftRouter Architecture,” HotNets, 2004.

[13] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, “A Blueprint for Introducing Disruptive Technology into the Internet,” *HotNets*, Oct. 2002.

[14] Larry Peterson, Scott Shenker, and Jon Turner, “Overcoming the Internet Impasse through Virtualization,” *HotNets*, Nov. 2004.

[15] Ram Keralapura, Nina Taft, Chen-Nee Chuah, and Gianluca Iannaccone, “Can ISPs take the heat from Overlay Networks?,” *HotNets*, Nov. 2004.

[16] C. Perkins, ”IP Mobility Support for IPv4,” RFC3344, August 2002.

[17] E. Rosen and Y. Rekhter, ”BGP/MPLS VPNs,” RFC2547, March 1999.

[18] L. Yang, R. Dantu, T. Anderson, R. Gopal, ”Forwarding and Control Element Separation (ForCES) Framework,” RFC 3746, April 2004.