Virtual Routers on the Move: Live Router Migration as a Network-Management Primitive

Yi Wang* Eric Keller* Brian Biskeborn* Jacobus van der Merwe† Jennifer Rexford*

* Princeton University, Princeton, NJ, USA † AT&T Labs - Research, Florham Park, NJ, USA

{yiwang,jrex}@cs.princeton.edu {ekeller,bbiskebo}@princeton.edu kobus@research.att.com

ABSTRACT

The complexity of network management is widely recognized as one of the biggest challenges facing the Internet today. Point solutions for individual problems further increase system complexity while not addressing the underlying causes. In this paper, we argue that many network-management problems stem from the same root cause—the need to maintain consistency between the physical and logical configuration of the routers. Hence, we propose VROOM (Virtual ROuters On the Move), a new network-management primitive that avoids unnecessary changes to the logical topology by allowing (virtual) routers to freely move from one physical node to another. In addition to simplifying existing network-management tasks like planned maintenance and service deployment, VROOM can also help tackle emerging challenges such as reducing energy consumption. We present the design, implementation, and evaluation of novel migration techniques for virtual routers with either hardware or software data planes. Our evaluation shows that VROOM is transparent to routing protocols and results in no performance impact on the data traffic when a hardware-based data plane is used.

Categories and Subject Descriptors

C.2.6 [**Computer Communication Networks**]: Internetworking; C.2.1 [**Computer Communication Networks**]: Network Architecture and Design

General Terms

Design, Experimentation, Management, Measurement

Keywords

Internet, architecture, routing, virtual router, migration

1. INTRODUCTION

Network management is widely recognized as one of the most important challenges facing the Internet. The cost of the people and systems that manage a network typically exceeds the cost of the underlying nodes and links; in addition, most network outages are caused by operator errors, rather than equipment failures [21]. From routine tasks such as planned maintenance to the less-frequent deployment of new protocols, network operators struggle to provide seamless service in the face of changes to the underlying network. Handling change is difficult because each change to the physical infrastructure requires a corresponding modification to the logical configuration of the routers—such as reconfiguring the tunable parameters in the routing protocols.

Logical refers to IP packet-forwarding functions, while *physical* refers to the physical router equipment (such as line cards and the CPU) that enables these functions. Any inconsistency between the logical and physical configurations can lead to unexpected reachability or performance problems. Furthermore, because of today's tight coupling between the physical and logical topologies, sometimes logical-layer changes are used purely as a *tool* to handle physical changes more gracefully. A classic example is increasing the link weights in Interior Gateway Protocols to "cost out" a router in advance of planned maintenance [30]. In this case, a change in the logical topology is *not* the goal, rather it is the indirect tool available to achieve the task at hand, and it does so with potential negative side effects.

In this paper, we argue that breaking the tight coupling between physical and logical configurations can provide a *single*, general abstraction that simplifies network management. Specifically, we propose VROOM (Virtual ROuters On the Move), a new network-management primitive where virtual routers can move freely from one physical router to another. In VROOM, physical routers merely serve as the carrier substrate on which the actual virtual routers operate. VROOM can migrate a virtual router to a different physical router without disrupting the flow of traffic or changing the logical topology, obviating the need to reconfigure the virtual routers while also avoiding routing-protocol convergence delays. For example, if a physical router must undergo planned maintenance, the virtual routers could move (in advance) to another physical router in the same Point-of-Presence (PoP). In addition, edge routers can move from one location to another by virtually re-homing the links that connect to neighboring domains.

Realizing these objectives presents several challenges: (i) *migratable routers*: to make a (virtual) router migratable, its "router" functionality must be separable from the physical equipment on which it runs; (ii) *minimal outages:* to avoid disrupting user traffic or triggering routing protocol reconvergence, the migration should cause no or minimal packet loss; (iii) *migratable links:* to keep the IP-layer topology in-

SIGCOMM'08, August 17–22, 2008, Seattle, Washington, USA.

tact, the links attached to a migrating router must "follow" it to its new location. Fortunately, the third challenge is addressed by recent advances in transport-layer technologies, as discussed in Section 2. Our goal, then, is to migrate router functionality from one piece of equipment to another without disrupting the IP-layer topology or the data traffic it carries, and without requiring router reconfiguration.

On the surface, virtual router migration might seem like a straight-forward extention to existing virtual machine migration techniques. This would involve copying the virtual router image (including routing-protocol binaries, configuration files and data-plane state) to the new physical router and freezing the running processes before copying them as well. The processes and data-plane state would then be restored on the new physical router and associated with the migrated links. However, the delays in completing all of these steps would cause unacceptable disruptions for both the data traffic and the routing protocols. For virtual router migration to be viable in practice, packet forwarding should not be interrupted, not even temporarily. In contrast, the control plane can tolerate brief disruptions, since routing protocols have their own retransmission mechansisms. Still, the control plane must restart quickly at the new location to avoid losing protocol adjacencies with other routers and to minimize delay in responding to unplanned network events.

In VROOM, we minimize disruption by leveraging the separation of the control and data planes in modern routers. We introduce a *data-plane hypervisor*—a migration-aware interface between the control and data planes. This unified interface allows us to support migration between physical routers with different data-plane technologies. VROOM migrates only the control plane, while continuing to forward traffic through the old data plane. The control plane can start running at the new location, and populate the new data plane while updating the old data plane in parallel. During the transition period, the old router redirects routing-protocol traffic to the new location. Once the data plane is fully populated at the new location, link migration can begin. The two data planes operate simultaneously for a period of time to facilitate asynchronous migration of the links.

To demonstrate the generality of our data-plane hypervisor, we present two prototype VROOM routers—one with a software data plane (in the Linux kernel) and the other with a hardware data plane (using a NetFPGA card [23]). Each virtual router runs the Quagga routing suite [26] in an OpenVZ container [24]. Our software extensions consist of three main modules that (i) separate the forwarding tables from the container contexts, (ii) push the forwarding-table entries generated by Quagga into the separate data plane, and (iii) dynamically bind the virtual interfaces and forwarding tables. Our system supports seamless live migration of virtual routers between the two data-plane platforms. Our experiments show that virtual router migration causes no packet loss or delay when the hardware data plane is used, and at most a few seconds of delay in processing control-plane messages.

The remainder of the paper is structured as follows. Section 2 presents background on flexible transport networks and an overview of related work. Next, Section 3 discusses how router migration would simplify existing network management tasks, such as planned maintenance and service deployment, while also addressing emerging challenges like

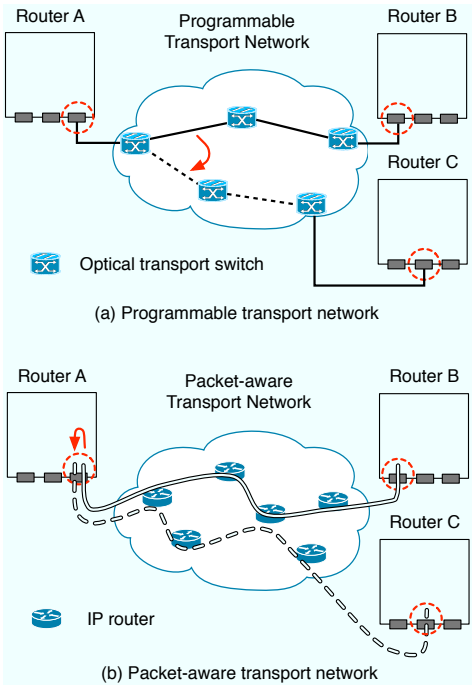

Figure 1: Link migration in the transport networks

power management. We present the VROOM architecture in Section 4, followed by the implementation and evaluation in Sections 5 and 6, respectively. We briefly discuss our on-going work on migration scheduling in Section 7 and conclude in Section 8.

2. BACKGROUND

One of the fundamental requirements of VROOM is "link migration", i.e., the links of a virtual router should "follow" its migration from one physical node to another. This is made possible by emerging transport network technologies. We briefly describe these technologies before giving an overview of related work.

2.1 Flexible Link Migration

In its most basic form, a link at the IP layer corresponds to a direct physical link (e.g., a cable), making link migration hard as it involves physically moving link end point(s). However, in practice, what appears as a direct link at the IP layer often corresponds to a series of connections through different network elements at the transport layer. For example, in today's ISP backbones, "direct" physical links are typically realized by optical transport networks, where an IP link corresponds to a circuit traversing multiple optical switches [9, 34]. Recent advances in *programmable transport networks* [9, 3] allow physical links between routers to be dynamically set up and torn down. For example, as shown in Figure 1(a), the link between physical routers A and B is switched through a programmable transport network. By signaling the transport network, the same physical port on router A can be connected to router C after an optical path switch-over. Such path switch-over at the transport layer can be done efficiently, e.g., sub-nanosecond optical switching time has been reported [27]. Furthermore, such switching can be performed across a wide-area network of transport switches, which enables inter-POP link migration.

In addition to *core links* within an ISP, we also want to migrate *access links* connecting customer edge (CE) routers and provider edge (PE) routers, where only the PE end of the links are under the ISP's control. Historically, access links correspond to a path in the underlying access network, such as a T1 circuit in a time-division multiplexing (TDM) access network. In such cases, the migration of an access link can be accomplished in similar fashion to the mechanism shown in Figure 1(a), by switching to a new circuit at the switch directly connected to the CE router. However, in traditional circuit-switched access networks, a dedicated physical port on a PE router is required to terminate each TDM circuit. Therefore, if all ports on a physical PE router are in use, it will not be able to accommodate more virtual routers. Fortunately, as Ethernet emerges as an economical and flexible alternative to legacy TDM services, access networks are evolving to *packet-aware* transport networks [2]. This trend offers important benefits for VROOM by eliminating the need for per-customer physical ports on PE routers. In a packet-aware access network (e.g., a virtual private LAN service access network), each customer access port is associated with a label, or a "pseudo wire" [6], which allows a PE router to support multiple logical access links on the same physical port. The migration of a pseudo-wire access link involves establishing a new pseudo wire and switching to it at the multi-service switch [2] adjacent to the CE.

Unlike conventional ISP networks, some networks are realized as overlays on top of other ISPs' networks. Examples include commercial "Carrier Supporting Carrier (CSC)" networks [10], and VINI, a research virtual network infrastructure overlaid on top of National Lambda Rail and Internet2 [32]. In such cases, a single-hop link in the overlay network is actually a multi-hop path in the underlying network, which can be an MPLS VPN (e.g., CSC) or an IP network (e.g., VINI). Link migration in an MPLS transport network involves switching over to a newly established label switched path (LSP). Link migration in an IP network can be done by changing the IP address of the tunnel end point.

2.2 Related Work

VROOM's motivation is similar, in part, to that of the RouterFarm work [3], namely, to reduce the impact of planned maintenance by migrating router functionality from one place in the network to another. However, RouterFarm essentially performs a "cold restart", compared to VROOM's live ("hot") migration. Specifically, in RouterFarm router migration is realized by re-instantiating a router instance at the new location, which not only requires router reconfiguration, but also introduces inevitable downtime in both the control and data planes. In VROOM, on the other hand, we perform *live* router migration without reconfiguration or discernible disruption. In our earlier prototype of VROOM [33], router migration was realized by directly using the standard virtual machine migration capability provided by Xen [4], which lacked the control and data plane separation presented in this paper. As a result, it involved data-plane downtime during the migration process.

Recent advances in virtual machine technologies and their live migration capabilities [12, 24] have been leveraged in server-management tools, primarily in data centers. For example, Sandpiper [35] automatically migrates virtual servers across a pool of physical servers to alleviate hotspots. Usher [22] allows administrators to express a variety of policies for managing clusters of virtual servers. Remus [13] uses asynchronous virtual machine replication to provide high availability to server in the face of hardware failures. In contrast, VROOM focuses on leveraging live migration techniques to simplify management in the networking domain.

Network virtualization has been proposed in various contexts. Early work includes the "switchlets" concept, in which ATM switches are partitioned to enable dynamic creation of virtual networks [31]. More recently, the CABO architecture proposes to use virtualization as a means to enable multiple service providers to share the same physical infrastructure [16]. Outside the research community, router virtualization has already become available in several forms in commercial routers [11, 20]. In VROOM, we take an additional step not only to virtualize the router functionality, but also to decouple the virtualized router from its physical host and enable it to migrate.

VROOM also relates to recent work on minimizing transient routing disruptions during planned maintenance. A measurement study of a large ISP showed that more than half of routing changes were planned in advance [19]. Network operators can limit the disruption by reconfiguring the routing protocols to direct traffic away from the equipment undergoing maintenance [30, 17]. In addition, extensions to the routing protocols can allow a router to continue forwarding packets in the data plane while reinstalling or rebooting the control-plane software [29, 8]. However, these techniques require changes to the logical configuration or the routing software, respectively. In contrast, VROOM hides the effects of physical topology changes in the first place, obviating the need for point solutions that increase system complexity while enabling new network-management capabilities, as discussed in the next section.

3. NETWORK MANAGEMENT TASKS

In this section, we present three case studies of the applications of VROOM. We show that the separation between physical and logical, and the router migration capability enabled by VROOM, can greatly simplify existing network-management tasks. It can also provide network-management solutions to other emerging challenges. We explain why the existing solutions (in the first two examples) are not satisfactory and outline the VROOM approach to addressing the same problems.

3.1 Planned Maintenance

Planned maintenance is a hidden fact of life in every network. However, the state-of-the-art practices are still unsatisfactory. For example, software upgrades today still require rebooting the router and re-synchronizing routing protocol states from neighbors (e.g., BGP routes), which can lead to outages of 10-15 minutes [3]. Different solutions have been proposed to reduce the impact of planned maintenance on network traffic, such as "costing out" the equipment in advance. Another example is the RouterFarm approach of removing the static binding between customers and access routers to reduce service disruption time while performing maintenance on access routers [3]. However, we argue that neither solution is satisfactory, since maintenance of *physical* routers still requires changes to the *logical* network topology, and requires (often human interactive) reconfigurations and routing protocol reconvergence. This usually implies more configuration errors [21] and increased network instability.

We performed an analysis of planned-maintenance events conducted in a Tier-1 ISP backbone over a one-week period. Due to space limitations, we only mention the high-level results that are pertinent to VROOM here. Our analysis indicates that, among all the planned-maintenance events that have undesirable network impact today (e.g., routing protocol reconvergence or data-plane disruption), 70% could be conducted without any network impact if VROOM were used. (This number assumes migration between routers with control planes of like kind. With more sophisticated migration strategies, e.g., where a "control-plane hypervisor" allows migration between routers with different control plane implementations, the number increases to 90%.) These promising numbers result from the fact that most planned-maintenance events were hardware related and, as such, did not intend to make any longer-term changes to the logical-layer configurations.

To perform planned maintenance tasks in a VROOM-enabled network, network administrators can simply migrate all the virtual routers running on a physical router to other physical routers before doing maintenance and migrate them back afterwards as needed, without ever needing to reconfigure any routing protocols or worry about traffic disruption or protocol reconvergence.

3.2 Service Deployment and Evolution

Deploying new services, like IPv6 or IPTV, is the lifeblood of any ISP. Yet, ISPs must exercise caution when deploying these new services. First, they must ensure that the new services do not adversely impact existing services. Second, the necessary support systems need to be in place before services can be properly supported. (Support systems include configuration management, service monitoring, provisioning, and billing.) Hence, ISPs usually start with a small trial running in a controlled environment on dedicated equipment, supporting a few early-adopter customers. However, this leads to a "success disaster" when the service warrants wider deployment. The ISP wants to offer seamless service to its existing customers, and yet also restructure their test network, or move the service onto a larger network to serve a larger set of customers. This "trial system success" dilemma is hard to resolve if the *logical* notion of a "network node" remains bound to a specific *physical* router.

VROOM provides a simple solution by enabling network operators to freely migrate virtual routers from the trial system to the operational backbone. Rather than shutting down the trial service, the ISP can continue supporting the early-adopter customers while continuously growing the trial system, attracting new customers, and eventually seamlessly migrating the entire service to the operational network.

ISPs usually deploy such service-oriented routers as close to their customers as possible, in order to avoid backhaul traffic. However, as the services grow, the geographical distribution of customers may change over time. With VROOM, ISPs can easily reallocate the routers to adapt to new customer demands.

3.3 Power Savings

VROOM not only provides simple solutions to conventional network-management tasks, but also enables new solutions to emerging challenges such as power management. It was reported that in 2000 the total power consumption of the estimated 3.26 million routers in the U.S. was about 1.1 TWh (Tera-Watt hours) [28]. This number was expected to grow to 1.9 to 2.4TWh in the year 2005 by three different projection models [28], which translates into an annual cost of about 178-225 million dollars [25]. These numbers do not include the power consumption of the required cooling systems.

Although designing energy-efficient equipment is clearly an important part of the solution [18], we believe that network operators can also *manage* a network in a more power-efficient manner. Previous studies have reported that Internet traffic has a consistent diurnal pattern caused by human interactive network activities. However, today's routers are surprisingly power-insensitive to the traffic loads they are handling—an idle router consumes over 90% of the power it requires when working at maximum capacity [7]. We argue that, with VROOM, the variations in daily traffic volume can be exploited to reduce power consumption. Specifically, the size of the physical network can be expanded and shrunk according to traffic demand, by hibernating or powering down the routers that are not needed. The best way to do this today would be to use the "cost-out/cost-in" approach, which inevitably introduces configuration overhead and performance disruptions due to protocol reconvergence.

VROOM provides a cleaner solution: as the network traffic volume decreases at night, virtual routers can be migrated to a smaller set of physical routers and the unneeded physical routers can be shut down or put into hibernation to save power. When the traffic starts to increase, physical routers can be brought up again and virtual routers can be migrated back accordingly. With VROOM, the IP-layer topology stays intact during the migrations, so that power savings do not come at the price of user traffic disruption, reconfiguration overhead or protocol reconvergence. Our analysis of data traffic volumes in a Tier-1 ISP backbone suggests that, even if only migrating virtual routers within the same POP while keeping the same link utilization rate, applying the above VROOM power management approach could save 18%-25% of the power required to run the routers in the network. As discussed in Section 7, allowing migration across different POPs could result in more substantial power savings.

4. VROOM ARCHITECTURE

In this section, we present the VROOM architecture. We first describe the three building-blocks that make virtual router migration possible—router virtualization, control and data plane separation, and dynamic interface binding. We then present the VROOM router migration process. Unlike regular servers, modern routers typically have physically separate control and data planes. Leveraging this unique property, we introduce a *data-plane hypervisor* between the control and data planes that enables virtual routers to migrate across different data-plane platforms. We describe in detail the three migration techniques that minimize control-plane downtime and eliminate data-plane disruption—data-plane cloning, remote control plane, and double data planes.

4.1 Making Virtual Routers Migratable

Figure 2 shows the architecture of a VROOM router that supports virtual router migration. It has three important features that make migration possible: router virtualization, control and data plane separation, and dynamic interface binding, all of which already exist in some form in today's

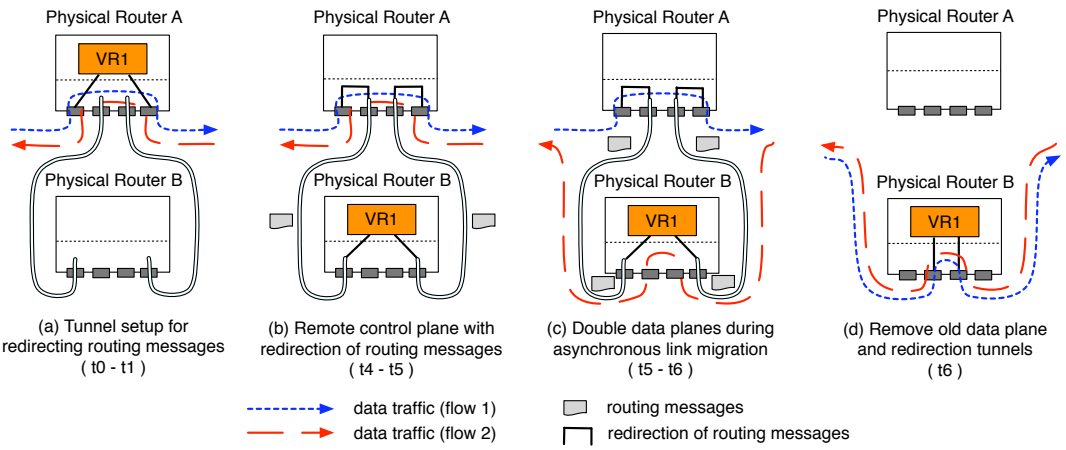

Figure 3: VROOM's novel router migration mechanisms (the times at the bottom of the subfigures correspond to those in Figure 4)

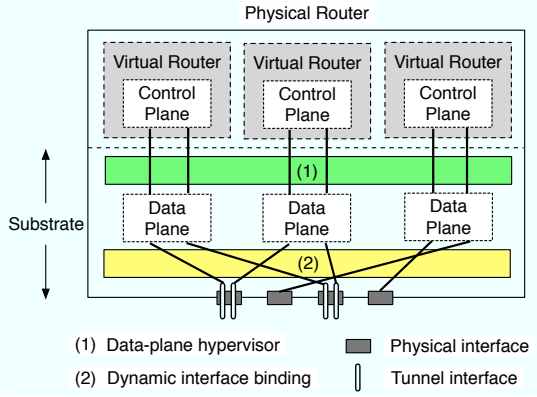

Figure 2: The architecture of a VROOM router

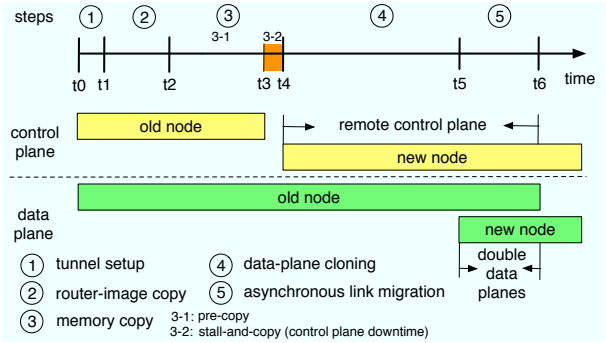

Figure 4: VROOM's router migration process

high-end commercial routers.

Router Virtualization: A VROOM router partitions the resources of a physical router to support multiple *virtual router* instances. Each virtual router runs independently with its own control plane (e.g., applications, configurations, routing protocol instances and routing information base (RIB)) and data plane (e.g., interfaces and forwarding information base (FIB)). Such *router virtualization* support is already available in some commercial routers [11, 20]. The isolation between virtual routers makes it possible to migrate one virtual router without affecting the others.

Control and Data Plane Separation: In a VROOM router, the control and data planes run in *separate* environments. As shown in Figure 2, the control planes of virtual routers are hosted in separate "containers" (or "virtual environments"), while their data planes reside in the *substrate*, where each data plane is kept in separate data structures with its own state information, such as FIB entries and access control lists (ACLs). Similar separation of control and data planes already exists in today's commercial routers, with control plane running on the CPU(s) and main memory, while the data plane runs on line cards that have their own computing power (for packet forwarding) and memory (to hold the FIBs). This separation allows VROOM to migrate the control and data planes of a virtual router separately (as discussed in Section 4.2.1 and 4.2.2).

Dynamic Interface Binding: To enable router migration and link migration, a VROOM router should be able to *dynamically* set up and change the binding between a virtual router's FIB and its *substrate interfaces* (which can be physical or tunnel interfaces), as shown in Figure 2. Given the existing interface binding mechanism in today's routers that maps interfaces with virtual routers, VROOM only requires two simple extensions. First, after a virtual router is migrated, this binding needs to be re-established dynamically on the new physical router. This is essentially the same as if this virtual router were just instantiated on the physical router. Second, link migration in a packet-aware transport network involves changing tunnel interfaces in the router, as shown in Figure 1. In this case, the router substrate needs to switch the binding from the old tunnel interface to the new one on-the-fly[1].

4.2 Virtual Router Migration Process

Figures 3 and 4 illustrate the VROOM virtual router migration process. The first step in the process involves establishing tunnels between the source physical router A and destination physical router B of the migration (Figure 3(a)). These tunnels allow the control plane to send and receive routing messages after it is migrated (steps 2 and 3) but before link migration (step 5) completes. They also allow the migrated control plane to keep its data plane on A up-to-

[1]In the case of a programmable transport network, link migration happens inside the transport network and is transparent to the routers.

date (Figure 3(b)). Although the control plane will experience a short period of downtime at the end of step 3 (memory copy), the data plane continues working during the entire migration process. In fact, after step 4 (data-plane cloning), the data planes on both A and B can forward traffic simultaneously (Figure 3(c)). With these double data planes, links can be migrated from A to B in an asynchronous fashion (Figure 3(c) and (d)), after which the data plane on A can be disabled (Figure 4). We now describe the migration mechanisms in greater detail.

4.2.1 Control-Plane Migration

Two things need to be taken care of when migrating the control plane: the *router image*, such as routing-protocol binaries and network configuration files, and the *memory*, which includes the states of all the running processes. When copying the router image and memory, it is desirable to minimize the total migration time, and more importantly, to minimize the control-plane downtime (i.e., the time between when the control plane is check-pointed on the source node and when it is restored on the destination node). This is because, although routing protocols can usually tolerate a brief network glitch using retransmission (e.g., BGP uses TCP retransmission, while OSPF uses its own reliable retransmission mechanism), a long control-plane outage can break protocol adjacencies and cause protocols to reconverge.

We now describe how VROOM leverages virtual machine (VM) migration techniques to migrate the control plane in steps 2 (router-image copy) and 3 (memory copy) of its migration process, as shown in Figure 4.

Unlike general-purpose VMs that can potentially be running completely different programs, virtual routers from the same vendor run the same (usually small) set of programs (e.g., routing protocol suites). VROOM assumes that the same set of binaries are already available on every physical router. Before a virtual router is migrated, the binaries are locally copied to its file system on the destination node. Therefore, only the router configuration files need to be copied over the network, reducing the total migration time (as local-copy is usually faster than network-copy).

The simplest way to migrate the memory of a virtual router is to check-point the router, copy the memory pages to the destination, and restore the router, a.k.a. *stall-and-copy* [24]. This approach leads to downtime that is proportional to the memory size of the router. A better approach is to add an iterative *pre-copy* phase before the final stall-and-copy [12], as shown in Figure 4. All pages are transferred in the first round of the pre-copy phase, and in the following rounds, only pages that were modified during the previous round are transferred. This pre-copy technique reduces the number of pages that need to be transfered in the stall-and-copy phase, reducing the control plane downtime of the virtual router (i.e., the control plane is only "frozen" between t3 and t4 in Figure 4).

4.2.2 Data-Plane Cloning

The control-plane migration described above could be extended to migrate the data plane, i.e., copy all data-plane states over to the new physical node. However, this approach has two drawbacks. First, copying the data-plane states (e.g., FIB and ACLs) is unnecessary and wasteful, because the information that is used to generate these states (e.g., RIB and configuration files) is already available in the control plane. Second, copying the data-plane state directly can be difficult if the source and destination routers use different data-plane technologies. For example, some routers may use TCAM (ternary content-addressable memory) in their data planes, while others may use regular SRAM. As a result, the data structures that hold the state may be different.

VROOM formalizes the interface between the control and data planes by introducing a *data-plane hypervisor*, which allows a migrated control plane to re-instantiate the data plane on the new platform, a process we call **data-plane cloning**. That is, only the control plane of the router is actually migrated. Once the control plane is migrated to the new physical router, it *clones* its original data plane by repopulating the FIB using its RIB and reinstalling ACLs and other data-plane states[2] through the data-plane hypervisor (as shown in Figure 2). The data-plane hypervisor provides a unified interface to the control plane that hides the heterogeneity of the underlying data-plane implementations, enabling virtual routers to migrate between different types of data planes.

4.2.3 Remote Control Plane

As shown in Figure 3(b), after VR1's control plane is migrated from A to B, the natural next steps are to repopulate (clone) the data plane on B and then migrate the links from A to B. Unfortunately, the creation of the new data plane can not be done instantaneously, primarily due to the time it takes to install FIB entries. Installing one FIB entry typically takes between one hundred and a few hundred microseconds [5]; therefore, installing the full Internet BGP routing table (about 250k routes) could take over 20 seconds. During this period of time, although data traffic can still be forwarded by the old data plane on A, all the routing instances in VR1's control plane can no longer send or receive routing messages. The longer the control plane remains unreachable, the more likely it will lose its protocol adjacencies with its neighbors.

To overcome this dilemma, A's substrate starts redirecting all the routing messages destined to VR1 to B at the end of the control-plane migration (time t4 in Figure 4). This is done by establishing a tunnel between A and B for each of VR1's substrate interfaces. To avoid introducing any additional downtime in the control plane, these tunnels are established before the control-plane migration, as shown in Figure 3(a). With this redirection mechanism, VR1's control plane not only can exchange routing messages with its neighbors, it can also act as the **remote control plane** for its old data plane on A and continue to update the old FIB when routing changes happen.

4.2.4 Double Data Planes

In theory, at the end of the data-plane cloning step, VR1 can switch from the old data plane on A to the new one on B by migrating all its links from A to B simultaneously. However, performing accurate synchronous link migration across all the links is challenging, and could significantly increase the complexity of the system (because of the need to implement a synchronization mechanism).

[2]Data dynamically collected in the old data plane (such as NetFlow) can be copied and merged with the new one. Other path-specific statistics (such as queue length) will be reset as the previous results are no longer meaningful once the physical path changes.

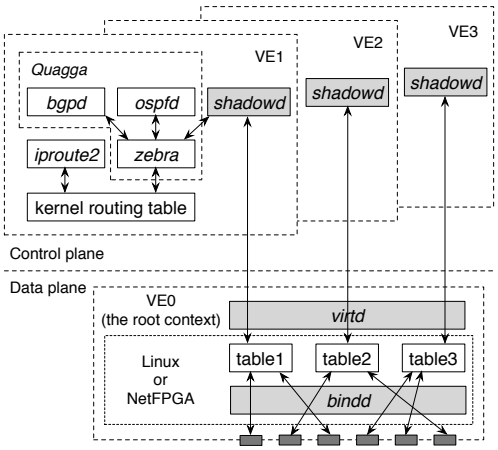

Figure 5: The design of the VROOM prototype routers (with two types of data planes)

Fortunately, because VR1 has **two** data planes ready to forward traffic at the end of the data-plane cloning step (Figure 4), the migration of its links does not need to happen all at once. Instead, each link can be migrated independent of the others, in an asynchronous fashion, as shown in Figure 3(c) and (d). First, router B creates a new *outgoing* link to each of VR1's neighbors, while all data traffic continues to flow through router A. Then, the *incoming* links can be safely migrated asynchronously, with some traffic starting to flow through router B while the remaining traffic still flows through router A. Finally, once all of VR1's links are migrated to router B, the old data plane and outgoing links on A, as well as the temporary tunnels, can be safely removed.

5. PROTOTYPE IMPLEMENTATION

In this section, we present the implementation of two VROOM prototype routers. The first is built on commodity PC hardware and the Linux-based virtualization solution OpenVZ [24]. The second is built using the same software but utilizing the NetFPGA platform [23] as the hardware data plane. We believe the design presented here is readily applicable to commercial routers, which typically have the same clean separation between the control and data planes.

Our prototype implementation consists of three new programs, as shown in Figure 5. These include `virtd`, to enable packet forwarding outside of the virtual environment (control and data plane separation); `shadowd`, to enable each VE to install routes into the FIB; and `bindd` (data plane cloning), to provide the bindings between the physical interfaces and the virtual interfaces and FIB of each VE (data-plane hypervisor). We first discuss the mechanisms that enable virtual router migration in our prototypes and then present the additional mechanisms we implemented that realize the migration.

5.1 Enabling Virtual Router Migration

We chose to use OpenVZ [24], a Linux-based OS-level virtualization solution, as the virtualization environment for our prototypes. As running multiple operating systems for different virtual routers is unnecessary, the lighter-weight OS-level virtualization is better suited to our need than other virtualization techniques, such as full virtualization and para-virtualization. In OpenVZ, multiple virtual environments (VEs) running on the same host share the same kernel, but have separate virtualized resources such as name spaces, process trees, devices, and network stacks. OpenVZ also provides live migration capability for running VEs[3].

In the rest of this subsection, we describe in a top-down order the three components of our two prototypes that enable virtual router migration. We first present the mechanism that separates the control and data planes, and then describe the data-plane hypervisor that allows the control planes to update the FIBs in the shared data plane. Finally, we describe the mechanisms that dynamically bind the interfaces with the FIBs and set up the data path.

5.1.1 Control and Data Plane Separation

To mimic the control and data plane separation provided in commercial routers, we move the FIBs out of the VEs and place them in a shared but virtualized data plane, as shown in Figure 5. This means that packet forwarding no longer happens within the context of each VE, so it is unaffected when the VE is migrated.

As previously mentioned, we have implemented two prototypes with different types of data planes—a software-based data plane (SD) and a hardware-based data plane (HD). In the SD prototype router, the data plane resides in the root context (or "VE0") of the system and uses the Linux kernel for packet forwarding. Since the Linux kernel (2.6.18) supports 256 separate routing tables, the SD router virtualizes its data plane by associating each VE with a different kernel routing table as its FIB.

In the HD router implementation, we use the NetFPGA platform configured with the reference router provided by Stanford [23]. The NetFPGA card is a 4-port gigabit ethernet PCI card with a Virtex 2-Pro FPGA on it. With the NetFPGA as the data plane, packet forwarding in the HD router does not use the host CPU, thus more closely resembling commercial router architectures. The NetFPGA reference router does not currently support virtualization. As a result, our HD router implementation is currently limited to only one virtual router per physical node.

5.1.2 Data-Plane Hypervisor

As explained in Section 4, VROOM extends the standard control plane/data plane interface to a migration-aware data-plane hypervisor. Our prototype presents a rudimentary data-plane hypervisor implementation which only supports FIB updates. (A full-fledged data-plane hypervisor would also allow the configuration of other data plane states.) We implemented the `virtd` program as the data-plane hypervisor. `virtd` runs in the VE0 and provides an interface for virtual routers to install/remove routes in the shared data plane, as shown in Figure 5. We also implemented the `shadowd` program that runs inside each VE and pushes route updates from the control plane to the FIB through `virtd`.

We run the Quagga routing software suite [26] as the control plane inside each VE. Quagga supports many routing protocols, including BGP and OSPF. In addition to the included protocols, Quagga provides an interface in `zebra`, its routing manager, to allow the addition of new protocol daemons. We made use of this interface to implement `shad-`

[3]The current OpenVZ migration function uses the simple "stall-and-copy" mechanism for memory migration. Including a "pre-copy" stage [12] in the process will reduce the migration downtime.

owd as a client of zebra. zebra provides clients with both the ability to notify zebra of route changes and to be notified of route changes. As shadowd is not a routing protocol but simply a shadowing daemon, it uses only the route redistribution capability. Through this interface, shadowd is notified of any changes in the RIB and immediately mirrors them to virtd using remote procedure calls (RPCs). Each shadowd instance is configured with a unique ID (e.g., the ID of the virtual router), which is included in every message it sends to virtd. Based on this ID, virtd can correctly install/remove routes in the corresponding FIB upon receiving updates from a shadowd instance. In the SD prototype, this involves using the Linux *iproute2* utility to set a routing table entry. In the HD prototype, this involves using the device driver to write to registers in the NetFPGA.

5.1.3 Dynamic Interface Binding

With the separation of control and data planes, and the sharing of the same data plane among multiple virtual routers, the data path of each virtual router must be set up properly to ensure that (i) data packets can be forwarded according to the right FIB, and (ii) routing messages can be delivered to the right control plane.

We implemented the bindd program that meets these requirements by providing two main functions. The first is to set up the mapping between a virtual router's substrate interfaces and its FIB after the virtual router is instantiated or migrated, to ensure correct packet forwarding. (Note that a virtual router's substrate interface could be either a dedicated physical interface or a tunnel interface that shares the same physical interface with other tunnels.) In the SD prototype, bindd establishes this binding by using the routing policy management function (i.e., "ip rule") provided by the Linux *iproute2* utility. As previously mentioned, the HD prototype is currently limited to a single table. Once NetFPGA supports virtualization, a mechanism similar to the "ip rule" function can be used to bind the interfaces with the FIBs.

The second function of bindd is to bind the substrate interfaces with the virtual interfaces of the control plane. In both prototypes, this binding is achieved by connecting each pair of substrate and virtual interfaces to a different bridge using the Linux *brctl* utility. In the HD prototype, each of the four physical ports on the NetFPGA is presented to Linux as a separate physical interface, so packets destined to the control plane of a local VE are passed from the NetFPGA to Linux through the corresponding interface.

5.2 Realizing Virtual Router Migration

The above mechanisms set the foundation for VROOM virtual router migration in the OpenVZ environment. We now describe the implementations of data-plane cloning, remote control plane, and double data planes.

Although migration is transparent to the routing processes running in the VE, shadowd needs to be notified at the end of the control plane migration in order to start the "data plane cloning". We implemented a function in shadowd that, when called, triggers shadowd to request zebra to resend all the routes and then push them down to virtd to repopulate the FIB. Note that virtd runs on a fixed (private) IP address and a fixed port on each physical node. Therefore, after a virtual router is migrated to a new physical node, the route updates sent by its shadowd can be seamlessly routed to the local virtd instance on the new node.

To enable a migrated control plane to continue updating the old FIB (i.e., to act as a "remote control plane"), we implemented in virtd the ability to forward route updates to another virtd instance using the same RPC mechanism that is used by shadowd. As soon as virtual router VR1 is migrated from node A to node B, the migration script notifies the virtd instance on B of A's IP address and VR1's ID. B's virtd, besides updating the new FIB, starts forwarding the route updates from VR1's control plane to A, whose virtd then updates VR1's old FIB. After all of VR1's links are migrated, the old data plane is no longer used, so B's virtd is notified to stop forwarding updates. With B's virtd updating both the old and new FIBs of VR1 (i.e., the "double data planes"), the two data planes can forward packets during the asynchronous link migration process.

Note that the data-plane hypervisor implementation makes the the control planes unaware of the details of a particular underlying data plane. As as result, migration can occur between any combination of our HD and SD prototypes (i.e. SD to SD, HD to HD, SD to HD, and HD to SD).

6. EVALUATION

In this section, we evaluate the performance of VROOM using our SD and HD prototype routers. We first measure the performance of the basic functions of the migration process individually, and then place a VROOM router in a network and evaluate the effect its migration has on the data and control planes. Specifically, we answer the following two questions:

1. *What is the impact of virtual router migration on data forwarding?* Our evaluation shows that it is important to have bandwidth isolation between migration traffic and data traffic. With separate bandwidth, migration based on an HD router has **no** performance impact on data forwarding. Migration based on a SD router introduces minimal delay increase and no packet loss to data traffic.

2. *What is the impact of virtual router migration on routing protocols?* Our evaluation shows that a virtual router running only OSPF in an Abilene-topology network can support 1-second OSPF *hello-interval* without losing protocol adjacencies during migration. The same router loaded with an additional full Internet BGP routing table can support a minimal OSPF *hello-interval* of 2 seconds without losing OSPF or BGP adjacencies.

6.1 Methodology

Our evaluation involved experiments conducted in the Emulab tesbed [15]. We primarily used PC3000 machines as the physical nodes in our experiments. The PC3000 is an Intel Xeon 3.0 GHz 64-bit platform with 2GB RAM and five Gigabit Ethernet NICs. For the HD prototype, each physical node was additionally equipped with a NetFPGA card. All nodes in our experiments were running an OpenVZ patched Linux kernel 2.6.18-ovz028stab049.1. For a few experiments we also used the lower performance PC850 physical nodes, built on an Intel Pentium III 850MHz platform with 512MB RAM and five 100Mbps Ethernet NICs.

We used three different testbed topologies in our experiments:

The diamond testbed: We use the 4-node diamond-topology testbed (Figure 6) to evaluate the performance of individual migration functions and the impact of migration on the data

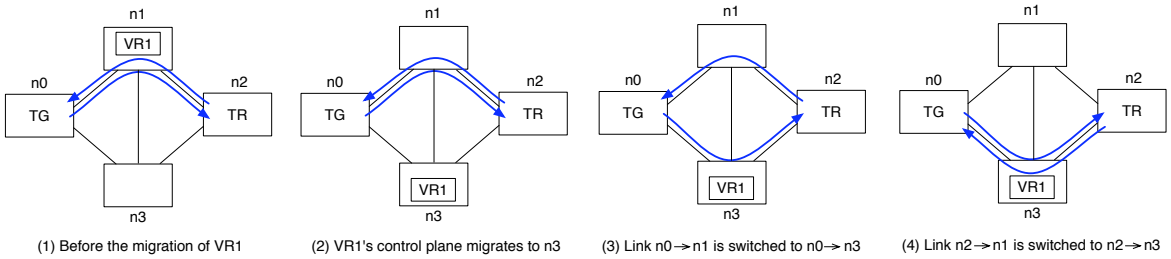

Figure 6: The diamond testbed and the experiment process

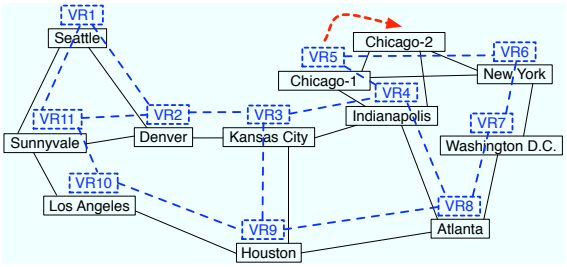

Figure 7: The Abilene testbed

Table 1: The memory dump file size of virtual router with different numbers of OSPF routes

Routes	0	10k	100k	200k	300k	400k	500k
Size (MB)	3.2	24.2	46.4	58.4	71.1	97.3	124.1

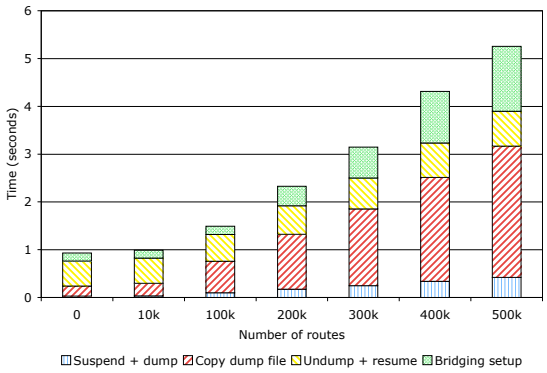

Figure 8: Virtual router memory-copy time with different numbers of routes

plane. The testbed has two different configurations, which have the same type of machines as physical node n0 and n2, but differ in the hardware on node n1 and n3. In the *SD* configuration, n1 and n3 are regular PCs on which we install our SD prototype routers. In the *HD* configuration, n1 and n3 are PCs each with a NetFPGA card, on which we install our HD prototype routers. In the experiments, virtual router VR1 is migrated from n1 to n3 through link n1→n3.

The dumbbell testbed: We use a 6-node dumbbell-shaped testbed to study the bandwidth contention between migration traffic and data traffic. In the testbed, round-trip UDP data traffic is sent between a pair of nodes while a virtual router is being migrated between another pair of nodes. The migration traffic and data traffic are forced to share the same physical link.

The Abilene testbed: We use a 12-node testbed (Figure 7) to evaluate the impact of migration on the control plane. It has a topology similar to the 11-node Abilene network backbone [1]. The only difference is that we add an additional physical node (Chicago-2), to which the virtual router on Chicago-1 (V5) is migrated. Figure 7 shows the initial topology of the virtual network, where 11 virtual routers (V1 to V11) run on the 11 physical nodes (except Chicago-2) respectively.

6.2 Performance of Migration Steps

In this subsection, we evaluate the performance of the two main migration functions of the prototypes—memory copy and FIB repopulation.

Memory copy: To evaluate memory copy time relative to the memory usage of the virtual router, we load the `ospfd` in VR1 with different numbers of routes. Table 1 lists the respective memory dump file sizes of VR1. Figure 8 shows the total time it takes to complete the memory-copy step, including (1) suspend/dump VR1 on n1, (2) copy the dump file from n1 to n3, (3) resume VR1 on n3, and (4) set up the bridging (interface binding) for VR1 on n3. We observe that as the number of routes becomes larger, the time it takes to copy the dump file becomes the dominating factor of the total memory copy time. We also note that when the memory usage becomes large, the bridging setup time also grows significantly. This is likely due to CPU contention with the virtual router restoration process, which happens at the same time.

FIB repopulation: We now measure the time it takes VR1 to repopulate the new FIB on n3 after its migration. In this experiment, we configure the virtual router with different numbers of static routes and measure the time it takes to install all the routes into the FIB in the software or hardware data plane. Table 2 compares the FIB update time and total time for FIB repopulation. FIB update time is the time `virtd` takes to install route entries into the FIB, while total time also includes the time for `shadowd` to send the routes to `virtd`. Our results show that installing a FIB entry into the NetFPGA hardware (7.4 microseconds) is over 250 times faster than installing a FIB entry into the Linux kernel routing table (1.94 milliseconds). As can be expected the update time increases linearly with the number of routes.

6.3 Data Plane Impact

In this subsection, we evaluate the influence router migration has on data traffic. We run our tests in both the HD and SD cases and compare the results. We also study the importance of having bandwidth isolation between the migration and data traffic.

Table 2: The FIB repopulating time of the SD and HD prototypes

Data plane type	Software data plane (SD)				Hardware data plane (HD)			
Number of routes	100	1k	10k	15k	100	1k	10k	15k
FIB update time (sec)	0.1946	1.9318	19.3996	31.2113	0.0008	0.0074	0.0738	0.1106
Total time (sec)	0.2110	2.0880	20.9851	33.8988	0.0102	0.0973	0.9634	1.4399

6.3.1 *Zero impact: HD router with separate migration bandwidth*

We first evaluate the data plane performance impact of migrating a virtual router from our HD prototype router. We configure the HD testbed such that the migration traffic from n1 to n3 goes through the direct link n1→n3, eliminating any potential bandwidth contention between the migration traffic and data traffic.

We run the D-ITG traffic generator [14] on n0 and n2 to generate round-trip UDP traffic. Our evaluation shows that, even with the maximum packet rate the D-ITG traffic generator on n0 can handle (sending and receiving 64-byte UDP packets at 91k packets/s), migrating the virtual router VR1 from n1 to n3 (including the control plane migration and link migration) does not have any performance impact on the data traffic it is forwarding—there is no delay increase or packet loss[4]. These results are not surprising, as the packet forwarding is handled by the NetFPGA, whereas the migration is handled by the CPU. This experiment demonstrates that hardware routers with separate migration bandwidth can migrate virtual routers with zero impact on data traffic.

6.3.2 *Minimal impact: SD router with separate migration bandwidth*

In the SD router case, CPU is the resource that could potentially become scarce during migration, because the control plane and data plane of a virtual router share the same CPU. We now study the case in which migration and packet forwarding together saturate the CPU of the physical node. As with the HD experiments above, we use link n1→n3 for the migration traffic to eliminate any bandwidth contention.

In order to create a CPU bottleneck on n1, we use PC3000 machines on n0 and n2 and use lower performance PC850 machines on n1 and n3. We migrate VR1 from n1 to n3 while sending round-trip UDP data traffic between nodes n0 and n2. We vary the packet rate of the data traffic from 1k to 30k packets/s and observe the performance impact the data traffic experiences due to the migration. (30k packets/s is the maximum bi-directional packet rate a PC850 machine can handle without dropping packets.)

Somewhat surprisingly, the delay increase caused by the migration is only noticeable when the packet rate is relatively low. When the UDP packet rate is at 5k packets/s, the control plane migration causes sporadic round-trip delay increases up to 3.7%. However, when the packet rate is higher (e.g., 25k packets/s), the change in delay during the migration is negligible ($< 0.4\%$).

This is because the packet forwarding is handled by kernel threads, whereas the OpenVZ migration is handled by user-level processes (e.g., `ssh`, `rsync`, etc.). Although kernel threads have higher priority than user-level processes in scheduling, Linux has a mechanism that prevents user-level processes from starving when the packet rate is high. This explains the delay increase when migration is in progress. However, the higher the packet rate is, the more frequently the user-level migration processes are interrupted, and more frequently the packet handler is called. Therefore, the higher the packet rate gets, the less additional delay the migration processes add to the packet forwarding. This explains why when the packet rate is 25k packets/s, the delay increase caused by migration becomes negligible. This also explains why migration does not cause any packet drops in the experiments. Finally, our experiments indicate that the link migration does not affect forwarding delay.

[4]We hard-wire the MAC addresses of adjacent interfaces on each physical nodes to eliminate the need for ARP request/response during link migration.

Table 3: Packet loss rate of the data traffic, with and without migration traffic

Data traffic rate (Mbps)	500	600	700	800	900
Baseline (%)	0	0	0	0	0.09
w/ migration traffic (%)	0	0	0.04	0.14	0.29

6.3.3 *Reserved migration bandwidth is important*

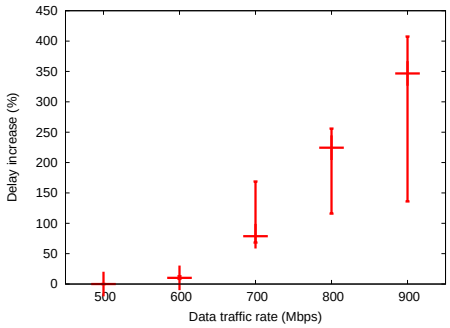

Figure 9: Delay increase of the data traffic, due to bandwidth contention with migration traffic

In 6.3.1 and 6.3.2, migration traffic is given its own link (i.e., has separate bandwidth). Here we study the importance of this requirement and the performance implications for data traffic if it is not met.

We use the dumbbell testbed in this experiment, where migration traffic and data traffic share the same bottleneck link. We load the `ospfd` of a virtual router with 250k routes. We start the data traffic rate from 500 Mbps, and gradually increase it to 900 Mbps. Because OpenVZ uses TCP (`scp`) for memory copy, the migration traffic only receives the left-over bandwidth of the UDP data traffic. As the available bandwidth decreases to below 300 Mbps, the migration time increases, which translates into a longer control-plane down-time for the virtual router.

Figure 9 compares the delay increase of the data traffic at different rates. Both the average delay and the delay jitter increase dramatically as the bandwidth contention becomes severe. Table 3 compares the packet loss rates of the data traffic at different rates, with and without migration

traffic. Not surprisingly, bandwidth contention (i.e., data traffic rate $\geq$ 700 Mbps) causes data packet loss. The above results indicate that in order to minimize the control-plane downtime of the virtual router, and to eliminate the performance impact to data traffic, operators should provide separate bandwidth for the migration traffic.

6.4 Control Plane Impact

In this subsection, we investigate the control plane dynamics introduced by router migration, especially how migration affects the protocol adjacencies. We assume a backbone network running MPLS, in which its edge routers run OSPF and BGP, while its core routers run only OSPF. Our results show that, with default timers, protocol adjacencies of both OSPF and BGP are kept intact, and at most one OSPF LSA retransmission is needed in the worst case.

6.4.1 Core Router Migration

We configure virtual routers VR1, VR6, VR8 and VR10 on the Abilene testbed (Figure 7) as edge routers, and the remaining virtual routers as core routers. By migrating VR5 from physical node Chicago-1 to Chicago-2, we observe the impact of migrating a core router on OSPF dynamics.

No events during migration: We first look at the case in which there are no network events during the migration. Our experiment results show that the control-plane downtime of VR5 is between 0.924 and 1.008 seconds, with an average of 0.972 seconds over 10 runs.

We start with the default OSPF timers of Cisco routers: *hello-interval* of 10 seconds and *dead-interval* of 40 seconds. We then reduce the *hello-interval* to 5, 2, and 1 second in subsequent runs, while keeping the *dead-interval* equal to four times the *hello-interval*. We find that the OSPF adjacencies between the migrating VR5 and its neighbors (VR4 and VR6) stay up in all cases. Even in the most restrictive 1-second *hello-interval* case, at most one OSPF hello message is lost and VR5 comes back up on Chicago-2 before its neighbors' dead timers expire.

Events happen during migration: We then investigate the case in which there are events during the migration and the migrating router VR5 misses the LSAs triggered by the events. We trigger new LSAs by flapping the link between VR2 and VR3. We observe that VR5 misses an LSA when the LSA is generated during VR5's 1-second downtime. In such a case, VR5 gets a retransmission of the missing LSA 5 seconds later, which is the default LSA *retransmit-interval*.

We then reduce the LSA *retransmit-interval* from 5 seconds to 1 second, in order to reduce the time that VR5 may have a stale view of the network. This change brings down the maximum interval between the occurrence of a link flap and VR5's reception of the resulting LSA to 2 seconds (i.e., the 1 second control plane downtime plus the 1 second LSA *retransmit-interval*).

6.4.2 Edge Router Migration

Here we configure VR5 as the fifth edge router in the network that runs BGP in addition to OSPF. VR5 receives a full Internet BGP routing table with 255k routes (obtained from RouteViewson Dec 12, 2007) from an eBGP peer that is not included in Figure 7, and it forms an iBGP full mesh with the other four edge routers.

With the addition of a full BGP table, the memory dump file size grows from 3.2 MB to 76.0 MB. As a result, it takes longer to suspend/dump the virtual router, copy over its dump file, and resume it. The average downtime of the control plane during migration increases to between 3.484 and 3.594 seconds, with an average of 3.560 seconds over 10 runs. We observe that all of VR5's BGP sessions stay intact during its migration. The minimal integer *hello-interval* VR5 can support without breaking its OSPF adjacencies during migration is 2 seconds (with *dead-interval* set to 8 seconds). In practice, ISPs are unlikely to set the timers much lower than the default values, in order to shield themselves from faulty links or equipment.

7. MIGRATION SCHEDULING

This paper primarily discusses the question of migration mechanisms ("how to migrate") for VROOM. Another important question is the migration scheduling ("where to migrate"). Here we briefly discuss the constraints that need to be considered when scheduling migration and several optimization problems that are part of our ongoing work on VROOM migration scheduling.

When deciding where to migrate a virtual router, several physical constraints need to be taken into consideration. First of all, an "eligible" destination physical router for migration must use a *software platform* compatible with the original physical router, and have similar (or greater) *capabilities* (such as the number of access control lists supported). In addition, the destination physical router must have sufficient resources available, including *processing power* (whether the physical router is already hosting the maximum number of virtual routers it can support) and *link capacity* (whether the links connected to the physical router have enough unused bandwidth to handle the migrating virtual router's traffic load). Furthermore, the *redundancy* requirement of the virtual router also needs to be considered—today a router is usually connected to two different routers (one as primary and the other as backup) for redundancy. If the primary and backup are migrated to the same node, physical redundancy will be lost.

Fortunately, ISPs typically leave enough "head room" in link capacities to absorb increased traffic volume. Additionally, most ISPs use routers from one or two vendors, with a small number of models, which leaves a large number of eligible physical routers to be chosen for the migration.

Given a physical router that requires maintenance, the question of where to migrate the virtual routers it currently hosts can be formulated as an optimization problem, subject to all the above constraints. Depending on the preference of the operator, different objectives can be used to pick the best destination router, such as minimizing the overall CPU load of the physical router, minimizing the maximum load of physical links in the network, minimizing the stretch (i.e., latency increase) of virtual links introduced by the migration, or maximizing the reliability of the network (e.g., the ability to survive the failure of any physical node or link). However, finding optimal solutions to these problems may be computationally intractable. Fortunately, simple local-search algorithms should perform reasonably well, since the number of physical routers to consider is limited (e.g., to hundreds or small thousands, even for large ISPs) and finding a "good" solution (rather than an optimal one) is acceptable in practice.

Besides migration scheduling for planned maintenance, we are also working on the scheduling problems of power sav-

ings and traffic engineering. In the case of power savings, we take the power prices in different geographic locations into account and try to minimize power consumption with a certain migration granularity (e.g., once every hour, according to the hourly traffic matrices). In the case of traffic engineering, we migrate virtual routers to shift load away from congested physical links.

8. CONCLUSIONS

VROOM is a new network-management primitive that supports live migration of virtual routers from one physical router to another. To minimize disruptions, VROOM allows the migrated control plane to clone the data-plane state at the new location while continuing to update the state at the old location. VROOM temporarily forwards packets using both data planes to support asynchronous migration of the links. These designs are readily applicable to commercial router platforms. Experiments with our prototype system demonstrate that VROOM does not disrupt the data plane and only briefly freezes the control plane. In the unlikely scenario that a control-plane event occurs during the freeze, the effects are largely hidden by existing mechanisms for retransmitting routing-protocol messages.

Our research on VROOM raises several broader questions about the design of future routers and the relationship with the underlying transport network. Recent innovations in transport networks support rapid set-up and tear-down of links, enabling the network topology to change underneath the IP routers. Dynamic topologies coupled with VROOM's migration of the control plane and cloning of the data plane make the router an increasingly ephemeral concept, not tied to a particular location or piece of hardware. Future work on router hypervisors could take this idea one step further. Just as today's commercial routers have a clear separation between the control and data planes, future routers could decouple the control-plane software from the control-plane state (e.g., routing information bases). Such a "control-plane hypervisor" would make it easier to upgrade router software and for virtual routers to migrate between physical routers that run different code bases.

9. REFERENCES

[1] The Internet2 Network. `http://www.internet2.edu/`.

[2] T. Afferton, R. Doverspike, C. Kalmanek, and K. K. Ramakrishnan. Packet-aware transport for metro networks. *IEEE Communication Magazine*, March 2004.

[3] M. Agrawal, S. Bailey, A. Greenberg, J. Pastor, P. Sebos, S. Seshan, J. van der Merwe, and J. Yates. RouterFarm: Towards a dynamic, manageable network edge. In *Proc. ACM SIGCOMM Workshop on Internet Network Management (INM)*, September 2006.

[4] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebar, I. Pratt, and A. Warfield. Xen and the Art of Virtualization. In *Proc. SOSP*, October 2003.

[5] O. Bonaventure, C. Filsfils, and P. Francois. Achieving sub-50 milliseconds recovery upon BGP peering link failures. *IEEE/ACM Trans. Networking*, October 2007.

[6] S. Bryant and P. Pate. Pseudo wire emulation edge-to-edge (PWE3) architecture. RFC 3985, March 2005.

[7] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright. Power awareness in network design and routing. In *Proc. IEEE INFOCOM*, 2008.

[8] E. Chen, R. Fernando, J. Scudder, and Y. Rekhter. Graceful Restart Mechanism for BGP. RFC 4724, January 2007.

[9] Ciena CoreDirector Switch. `http://www.ciena.com`.

[10] MPLS VPN Carrier Supporting Carrier. `http://www.cisco.com/en/US/docs/ios/12_0st/12_0st14/feature/guide/csc.html`.

[11] Cisco Logical Routers. `http://www.cisco.com/en/US/docs/ios_xr_sw/iosxr_r3.2/interfaces/command/reference/hr32lr.html`.

[12] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In *Proc. NSDI*, May 2005.

[13] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, N. Hutchinson, and A. Warfield. Remus: High availability via asynchronous virtual machine replication. In *Proc. NSDI*, April 2008.

[14] D-ITG. `http://www.grid.unina.it/software/ITG/`.

[15] Emulab. `http://www.emulab.net`.

[16] N. Feamster, L. Gao, and J. Rexford. How to lease the Internet in your spare time. *ACM SIGCOMM Computer Communications Review*, Jan 2007.

[17] P. Francois, M. Shand, and O. Bonaventure. Disruption-free topology reconfiguration in OSPF networks. In *Proc. IEEE INFOCOM*, May 2007.

[18] M. Gupta and S. Singh. Greening of the Internet. In *Proc. ACM SIGCOMM*, August 2003.

[19] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot. Feasibility of IP restoration in a tier-1 backbone. *IEEE Network Magazine*, Mar 2004.

[20] Juniper Logical Routers. `http://www.juniper.net/techpubs/software/junos/junos85/feature-guide-85/id-11139212.html`.

[21] Z. Kerravala. Configuration Management Delivers Business Resiliency. The Yankee Group, November 2002.

[22] M. McNett, D. Gupta, A. Vahdat, and G. M. Voelker. Usher: An extensible framework for managing clusters of virtual machines. In *Proc. USENIX LISA Conference*, November 2007.

[23] NetFPGA. `http://yuba.stanford.edu/NetFPGA/`.

[24] OpenVZ. `http://openvz.org`.

[25] Average retail price of electricity. `http://www.eia.doe.gov/cneaf/electricity/epm/table5_6_a.html`.

[26] Quagga Routing Suite. `http://www.quagga.net`.

[27] A. Rostami and E. Sargent. An optical integrated system for implementation of NxM optical cross-connect, beam splitter, mux/demux and combiner. IJCSNS International Journal of Computer Science and Network Security, July 2006.

[28] K. Roth, F. Goldstein, and J. Kleinman. Energy Consumption by Office and Telecommunications Equipment in commercial buildings Volume I: Energy Consumption Baseline. National Technical Information Service (NTIS), U.S. Department of Commerce, Springfield, VA 22161, NTIS Number: PB2002-101438, 2002.

[29] A. Shaikh, R. Dube, and A. Varma. Avoiding instability during graceful shutdown of multiple OSPF routers. *IEEE/ACM Trans. Networking*, 14(3):532–542, June 2006.

[30] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of hot-potato routing in IP networks. In *Proc. ACM SIGMETRICS*, June 2004.

[31] J. van der Merwe and I. Leslie. Switchlets and dynamic virtual ATM networks. In *Proc. IFIP/IEEE International Symposium on Integrated Network Management*, May 1997.

[32] VINI. `http://www.vini-veritas.net/`.

[33] Y. Wang, J. van der Merwe, and J. Rexford. VROOM: Virtual ROuters On the Move. In *Proc. ACM SIGCOMM Workshop on Hot Topics in Networking*, Nov 2007.

[34] J. Wei, K. Ramakrishnan, R. Doverspike, and J. Pastor. Convergence through packet-aware transport. *Journal of Optical Networking*, 5(4), April 2006.

[35] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif. Black-box and Gray-box Strategies for Virtual Machine Migration. In *Proc. NSDI*, April 2007.