

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]: Internet-networking; 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-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

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

Copyright 2008 ACM 978-1-60558-175-0/08/08 ...\$5.00.

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 extension 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 mechanisms. 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

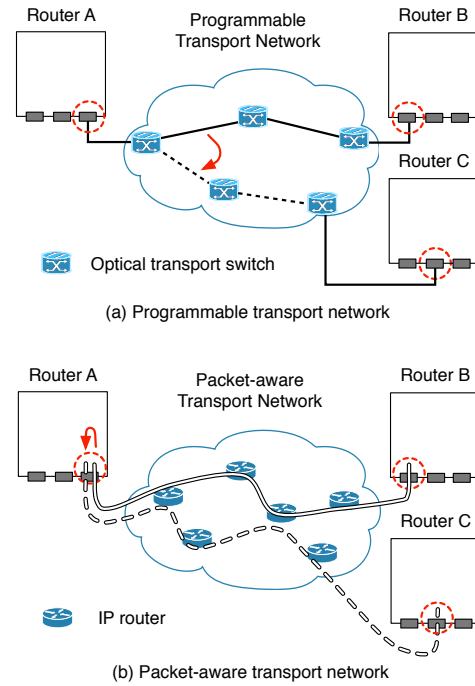


Figure 1: Link migration in the transport networks

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 convergence. 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 life-blood 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 [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 high-end commercial routers.

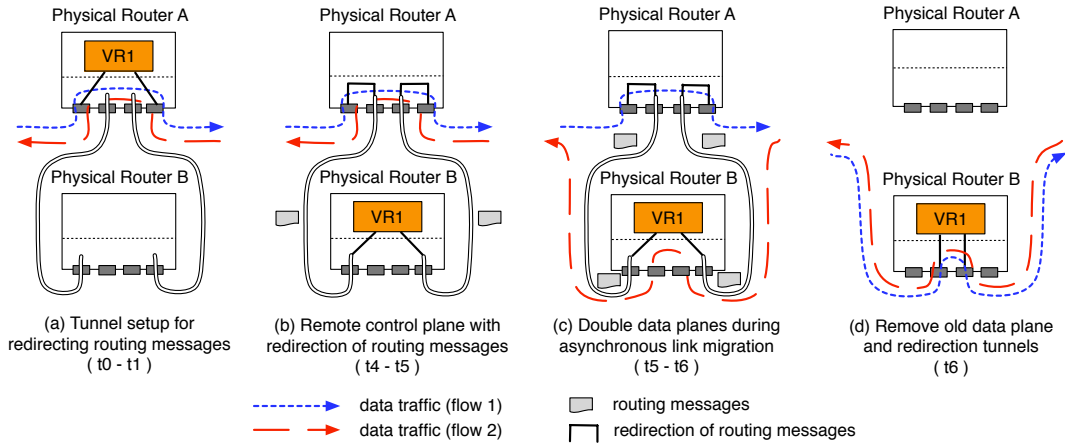


Figure 3: VROOM’s novel router migration mechanisms (the times at the bottom of the subfigures correspond to those in Figure 4)

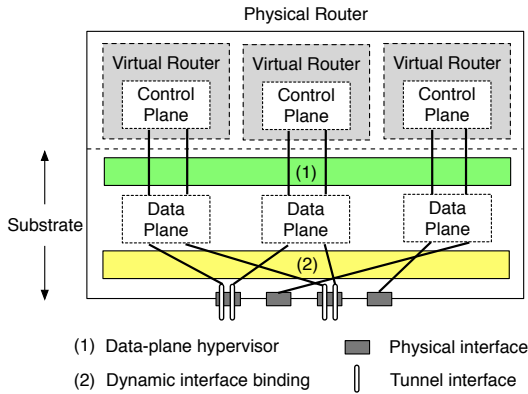


Figure 2: The architecture of a VROOM router

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 *dy-*

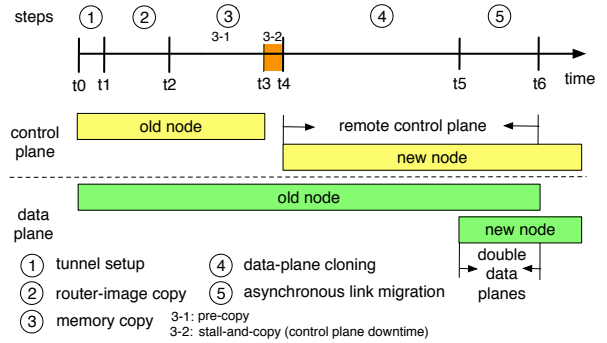


Figure 4: VROOM’s router migration process

namicallly 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-fly¹.

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-date (Figure 3(b)). Although the control plane will experi-

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

ence 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 transferred 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 states² 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).

Fortunately, because VR1 has **two** data planes ready to

²Data 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.

