Seamless BGP Migration With Router Grafting

Eric Keller
Princeton University

Jennifer Rexford
Princeton University

Jacobus van der Merwe
AT&T Labs - Research

Abstract

Network operators are under tremendous pressure to make their networks highly reliable to avoid service disruptions. Yet, operators often need to change the network to upgrade faulty equipment, deploy new services, and install new routers. Unfortunately, changes cause disruptions, forcing a trade-off between the benefit of the change and the disruption it will cause. In this paper we present *router grafting*, where parts of a router are seamlessly removed from one router and merged into another. We focus on grafting a BGP session and the underlying link–from one router to another, or between blades in a cluster-based router. Router grafting allows an operator to rehome a customer with no disruption, compared to downtimes today measured in minutes. In addition, grafting a BGP session can help in balancing load between routers or blades, planned maintenance, and even traffic management. We show that grafting a BGP session is practical even with today's monolithic router software. Our prototype implementation uses and extends Click, the Linux kernel, and Quagga, and introduces a daemon that automates the migration process.

1 Introduction

In nature, grafting is where a part of one living organism (e.g., tissue from a plant) is removed and fused into another organism. In this paper, we apply this concept to routers to enable new network-management capabilities which allow network changes to be made with minimal disruption. We call this *router grafting*. With router grafting, we view routers in terms of their parts and enable splitting these parts from one router and merging them into another. This capability makes the view of the network a more fluid one where the topology can readily change, allowing operators to adapt their networks without disruption in the service offered to users. We envision router grafting to eventually be applicable to arbitrary subsets of router resources and/or protocols. However, in this paper we take the first step towards this vision by focusing how to "graft" a BGP session and the underlying link from one router to another.

1.1 A Case for Router Grafting

The ability to adapt the network is an essential component of network management. Unfortunately, today's routers and routing protocols make change difficult. Changes to the network cause disruption, forcing operators to weigh the benefit of making a change against the potential impact performing the change will have. For example, today, the basic task of rehoming a BGP session requires shutting down the session, reconfiguring the new router, restarting the session, and exchanging a large amount of routing information typically leading to downtimes of several minutes. Further complicating matters is the fact that service-level agreements with customers often prohibit events that result in downtime without receiving prior approval and scheduling a maintenance window. This hand-cuffs the operator. In this section we provide several motivating examples of why seamless migration is needed and why it would be desirable to do at the level of individual sessions.

Load balancing across blades in a cluster router: Today's high-end routers have modular designs consisting of many cards—processor blades for running routing processes and interface cards for terminating links—spread over multiple chassis. In essence, the router itself is a large distributed system. Load balancing is an important function in distributed systems, and routers are no exception—today's routers often run near their limits of processing capacity [1]. Unfortunately, routers are not built with load balancing in mind. A BGP session is associated with a routing process on a particular blade upon establishment, making it difficult to shift load to another blade. A common approach used with Web servers is to drain load by directing new requests to other servers and waiting for existing requests to complete. Unfortunately, this technique is not applicable to routers, since routing sessions run indefinitely and unlike web services have persistent state. However, with the ability to migrate individual sessions, achieving better utilization of the router's processing capabilities is possible.

Rehoming a customer: An ISP homes a customer to a router based on geographic proximity and the availability of a router slot that can accommodate the customer's request [2]. However, this is done only at the time when a customer initiates service, based on the state of the network at that time. Rehoming might be necessary if the customer upgrades to a new service (such as multicast, IPv6, or advanced QoS or monitoring features) available only on a subset of routers. Rehoming is also necessary when an ISP upgrades or replaces a router and needs to move sessions from the old router to the new one. Customer rehoming involves moving the edge link—which can be done quickly because of recent innovations in layer-two access networks—as well as the BGP session.

Planned maintenance: Maintenance is a fact of life for network operators, yet, even though maintenance is planned in advance, little can be done to keep the router running. Consider a simple task of replacing a power supply. The best common practice is for operators to reconfigure the routing protocols to direct traffic away from that router and, once the traffic stops flowing, to take the router offline. Unfortunately, this approach only works for core routers within an ISP where alternate paths are available. At the edge of the network, an attractive alternative would be to graft all of the BGP sessions with neighboring networks to other routers to avoid disruptions in service. Migrating at the level of individual sessions is preferable to migrating all of the sessions and the routing processes as a group, since fine-grain migration allows multiple different routers to absorb only a small amount of extra load during the maintenance interval.

Traffic engineering: Traffic engineering is the act of reconfiguring the network to optimize the flow of traffic, to minimize congestion. Today, traffic engineering involves adjusting the routing-protocol parameters to coax the routers into computing new paths that better match the offered traffic, at the expense of transient disruptions during routing convergence. Router grafting enables a new approach to traffic engineering, where certain customers are rehomed to an edge router that better matches the traffic patterns. For example, if most of a customer's traffic leaves the ISP's network at a particular location, that customer could be rehomed closer to that egress point. In other words, we no longer need to consider the traffic matrix as fixed when performing traffic engineering—instead, we can change the traffic matrix to better match the backbone topology and routing by having traffic enter the network at a new location.

1.2 Challenges and Contributions

The benefits of router grafting are numerous. However, the design of today's routers and routing protocols make realizing router grafting challenging. Grafting a BGP session involves (i) migrating the underlying TCP connection, (ii) exchanging routing state, (iii) moving the routing-protocol configuration from one router to another, and (iv) migrating the underlying link. Ideally, all these actions need to be performed in a manner that is completely transparent (i.e., without involving the routers and operators in neighboring networks) and does not disrupt forwarding and routing (i.e., data packets are not dropped and routing adjacencies remain up).

Unfortunately, we cannot simply apply existing techniques for application-level session migration. Moving a BGP session to a different router changes the network topology and hence, the routing decisions at other routers. In particular, the remote end-point of the session must be informed of any routing changes—that is, any differences between the "best routes" chosen by the new and old homing points. Similarly, other routers in the ISP network need to change how they route toward destinations reachable through that remote end-point—they need to learn that these destinations are now reachable through the new homing location.

In addition, we cannot simply apply recently-proposed techniques for virtual-router migration [3], for two main reasons. First, the two physical routers may not be compatible—they may run different routing software (e.g., Cisco, Juniper, Quagga, or XORP). Second, we want to migrate and merge only a single BGP session, not the entire routing process, as many scenarios benefit from finer granularity. Instead, we view virtual-router migration as a complementary management primitive.

Fortunately, extending existing router software to support grafting requires only modest changes. The essential state that must be migrated is often well separated in the code. This makes it possible to export the state from one router and import it to another without much complexity. In this paper, we present an architecture for realizing router grafting and make the following contributions:

- Introduce the concept of router grafting, and realize an instance of it through BGP session migration. We demonstrate that BGP session migration can be performed in today's monolithic routing software, without much modification or refactoring of the code. Our fully-automated prototype router-grafting system is built by using and extending Click, Linux, and Quagga.

- Achieve transparency, where the remote BGP session end-point is not modified and is unaware migration is happening. We achieve this by bootstrap-

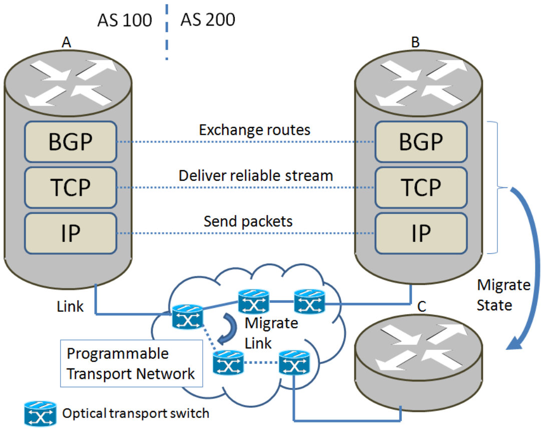


Figure 1: Migration protocol layers.

ping a routing session at the new homing location, with the old router emulating the remote end-point. The new homing point then takes over the role of the old router, sending the necessary routing updates to notify the remote end-point of routing changes.

- Introduce optimizations to nearly eliminate the impact of migration on other routers not directly involved in the migration. We achieve this by capitalizing on the fact that the routers already have much of the routing information they need, and that we know the identity of the old and new homing points.
- Describe an architecture where unplanned routing changes (such as link failures) during the grafting process do not affect correctness, and where packets are delivered successfully even during the migration. At worst, packets temporarily traverse a different path than the control plane advertises—a common situation during routing convergence.

The remainder of the paper is organized as follows. Section 2 discusses how the operation of BGP makes router grafting challenging. In Section 3 we present the router grafting architecture, focusing only on the control plane. Section 4 explains how we ensure correct routing and forwarding, even in the face of unplanned routing changes. In Section 5 we present our prototype, followed by a discussion of optimizations that reduce the overhead of grafting a BGP session in Section 6. We present an evaluation of our prototype and proposed optimizations in Section 7, followed by related work in Section 8 and the conclusion in Section 9.

2 BGP Routing Within a Single AS

Grafting a BGP session is difficult because BGP routing relies on many *layers* in the protocol stack and many *components* within an AS. In this section, we present a brief overview of BGP routing from the perspective of a single autonomous system (AS) to identify the challenges our grafting solution must address.

2.1 Protocol Layers: IP, TCP, & BGP

As illustrated in Figure 1, two neighboring routers exchange BGP update messages over a BGP session that runs on top of a TCP connection that, in turn, directs packets over the underlying IP link(s) between them. As such, grafting a BGP session will require moving the IP link, TCP connection, and BGP session from one location to another.

IP link: An AS connects to neighboring ASes through IP links. While a link could be a direct cable between two routers, these IP-layer links typically correspond to multiple hops in an underlying layer-two network. For example, routers at an exchange point often connect via a shared switch, and an ISP typically connects to its customers over an access network. These layer-two networks are increasingly programmable, allowing dynamic set-up and tear-down of layer-three links [4, 5, 6, 7]. This is illustrated in Figure 1 where the link between routers A and B is through a programmable transport network which can be changed to connect routers A and C. These innovations enable seamless migration of an IP link from one location to another within the scope of the layer-two network, such as rehoming a customer's access link to terminate on a different router in the ISP's network[1].

TCP connection: The neighboring routers exchange BGP messages over an underlying TCP connection. Unlike a conventional TCP connection between a Web client and a Web server, the connection must stay "up" for long periods of time, as the two routers are continuously exchanging messages. Further, each router sends keep-alive messages to enable the other router to detect lapses in connectivity. Upon missing three keep-alive messages, a router declares the other router as dead and discards all BGP routes learned from that neighbor. As such, grafting a BGP session requires timely migration of the underlying TCP connection.

BGP session: Two adjacent routers form a BGP session by first establishing a TCP session, then sending messages negotiating the properties of the BGP session, then exchanging the "best route" for each destination prefix. This process is controlled by a state machine that specifies what messages to exchange and how to handle them. Once the BGP session is established, the two

[1] Depending on the technology used to realize the layer-two network, the scope might be geographically contained, e.g., in the case of a packet access network, or might be significantly more spread out, e.g., in the case of a national footprint programmable optical transport network.

routers send incremental update messages—announcing new routes and withdrawing routes that are no longer available. A router stores the BGP routes learned from its neighbor in an *Adj-RIB-in* table, and the routes announced to the neighbor in an *Adj-RIB-out* table. Each BGP session has configuration state that controls how a router filters and modifies BGP routes that it imports from (or exports to) the remote neighbor. As such, grafting a BGP session requires transferring a large amount of RIB (Routing Information Base) state, as well as moving the associated configuration state.

2.2 Components: Blades, Routers, & ASes

A BGP session is associated with a routing process that runs on a processor blade within one of the routers in a larger AS. As such, grafting a BGP session involves extracting the necessary state from the routing process, transferring that state to another location, and changing the routing decisions at other routers as needed.

Processor blade: The simplest router has a processor for running the routing process, multiple interfaces for terminating links, and a switching fabric for directing packets from one interface to another. The BGP routing process maintains sessions with multiple neighbors and runs a decision process over the *Adj-RIB-in* tables to select a single "best" route for each destination prefix. The routing process stores the best route in a *Loc-RIB* table, and applies export policies to construct the *Adj-RIB-out* tables and send the corresponding update messages to each neighbor.

IP router: Today's high-end routers are large distributed systems, consisting of hundreds of interfaces and multiple processor blades spread over one or more chassis. These routers run multiple BGP processes—one on each processor blade—each responsible for a portion of the BGP sessions as shown in Figure 2. For a cluster-based router to scale, each BGP process runs its own decision process and exchanges its "best" route with the other BGP processes in the router, using a modified version of internal BGP (iBGP) [8]. This allows the distributed router to behave the same way as a simple router that runs a single BGP process. Any BGP process can handle any BGP session, since all processors can reach the interface cards through the switching fabric. As such, grafting a BGP session from one blade to another in the same router (e.g., the session with X from RP1 to RP2 in Figure 2) does not require migrating the underlying layer-three link.

Autonomous System (AS): An AS consists of multiple, geographically-distributed routers. Each router forms BGP sessions with neighboring routers in other ASes, and uses iBGP to disseminate its "best" route to other routers within the AS. The routers in the same AS also run an Interior Gateway Protocol (IGP), such as OSPF or IS-IS to compute paths to reach each other. Each router in the AS runs its own BGP process(es) and selects its own best route for each prefix. The routers may come to different decisions about the best route, not only because they learn different candidate routes but also because the decision depends on the IGP distances to other routers (in a practice known as hot-potato routing). This can be seen in Figure 3 where routers B and C have different paths to the destination *d*. As such, grafting a BGP session from one router to another (e.g., the session with A from router B to C in Figure 3) may change the BGP routing decisions.

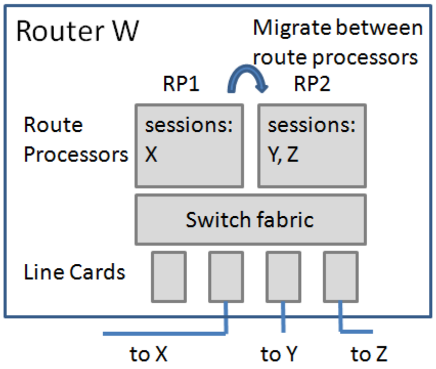


Figure 2: Migrating the session with X between route processor blades (from RP1 to RP2).

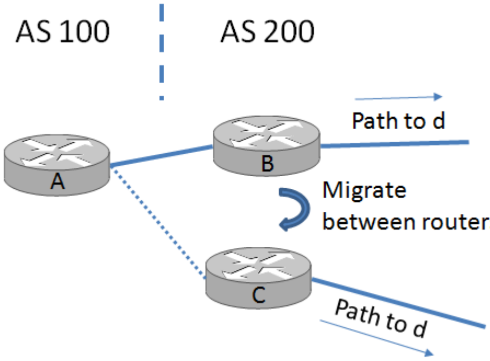


Figure 3: Migrating session with A between routers (from B to C).

3 Router Grafting Architecture

Seamless grafting of a BGP session relies on a careful progression through a number of coordinated steps. These steps are summarized in Figure 4, which shows a *migrate-from* router that hands off one of its BGP sessions to a *migrate-to* router in the same AS. These routers do not need to run the same software or be from the same vendor—they need only have the added support for router grafting. When the grafting process starts, the migrate-from router is responsible for handling a BGP session with the remote end-point router A (not shown). This BGP session with router A is to be migrated. The migrate-from router begins exporting the routing information and the migrate-to router is initialized with its own session-level data structures and a copy of the policy configuration, without actually establishing the session (Figure 4(a)). Then, the TCP connection is migrated, followed by the underlying link (Figure 4(b)). Finally, the migrate-to router imports the routing state and updates the other routers (Figure 4(c)), resulting in the migrate-to router handling the BGP session with the remote end-point ((Figure 4(d)). This section focuses exclusively on control-plane operations, deferring discussion of the data plane until Section 4.

3.1 Copying BGP Session Configuration

Each BGP session end-point has a variety of configuration state needed to establish the session with the remote end-point (with a given IP address and AS number) and apply policies for filtering and modifying route announcements. The network operators, or an automated management system, configure the session end-point by applying configuration commands at the router's command-line interface or uploading a new configuration file. The router stores the configuration information in various internal data structures.

Rather than exporting these internal data structures, we capitalize on the fact that the current configuration is captured in a well-defined format in the configuration file. Our design simply "dumps" the configuration file for the migrate-from router, extracts the commands relevant to the BGP session end-point, and applies these commands to the migrate-to router, after appropriate translation to account for vendor-dependent differences in the command syntax. This allows the migrate-to router to create its own internal data structures for the configuration information.

However, the migrate-to router is not yet ready to assume responsibility for the BGP session. To finish initializing the migrate-to router, we extend the BGP state machine to include an 'inactive' state, where the router can create data structures and import state for the session without attempting to communicate with the remote end-point. The migrate-to router transitions from the 'inactive' state to 'established' state when instructed by the grafting process.

3.2 Exporting & Resetting Run-Time State

A router maintains a variety of state for BGP session end-points. To meet our goals, BGP grafting need only consider the Routing Information Bases (RIBs)—the other state may be simply reinitialized at the migrate-to router[2].

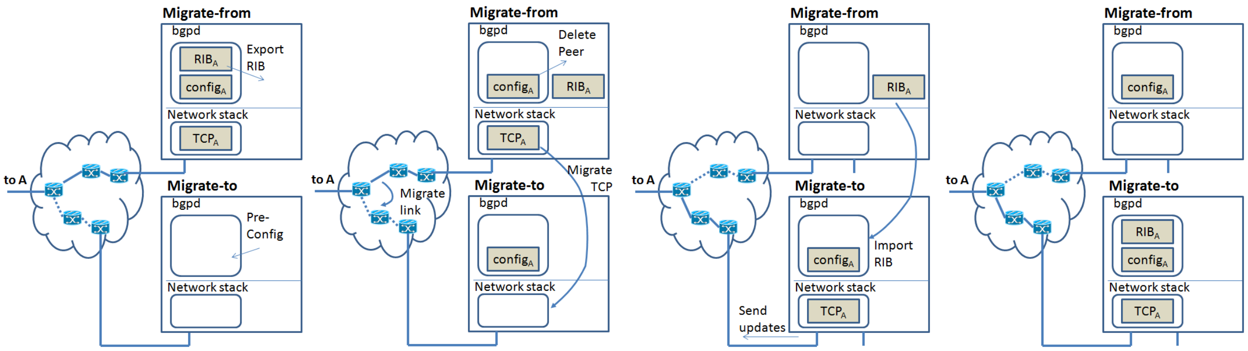
Routing Information Bases (RIBs): The most important state associated with the BGP session-end-point is stored in the routing information bases—the *Adj-RIB-in* and *Adj-RIB-out*. In our architecture, we dump the RIBs at the migrate-from router to prepare for importing the information at the migrate-to router. While the RIBs are represented differently on different router platforms, the information they store is standardized as part of the BGP protocol. In most router implementations, the RIB data structure is factored apart from the rest of the routing software, and many routers support commands for "dumping" the current RIBs. Even though the RIB dump formats vary by vendor, de facto standards like the popular MRT format [9] do exist.

State in the BGP state machine: A BGP session end-point stores information about the BGP state machine. We can forgo migrating this state – the BGP session is either 'established' or not. If the session is in one of the not-established states, we can simply close the session at the migrate-from router and start the migrate-to router in the idle state. This does not trigger any transient disruption—since the session is not "up" anyway. If the session at the migrate-from router is 'established,' we can start the new session at the migrate-to router in the 'inactive' state.

BGP timers: BGP implementations also include a variety of timers, many of which are vendor-dependent. For example, some routers use an MRAI (Minimum Route Advertisement Interval) timer to pace the transmission of BGP update messages. This is purely a local operation at one end-point of the session, not requiring any agreement with the remote end-point. Another common timer is the keep-alive interval that drives the periodic sending of heartbeat messages, and a hold timer for detecting missing keep-alive messages from the remote end-point. Fortunately, missing a single keep-alive message, or sending the message slightly early or late, would not erroneously detect a session failure because routers typically wait for *three* missed keep-alive messages before tearing down the session. As such, we do not migrate BGP timer values and instead simply initialize whatever timers are used at the migrate-to router.

BGP statistics: BGP implementations maintain numerous statistics about each session and even individual routes. These statistics, while broadly useful for network monitoring, are not essential to the correct operation of the router. They only have meaning at the local session

[2] Router grafting does not preclude the remaining state from being included, simply we chose not to in order to keep code modifications at a minimum while still meeting our goals of (i) routing protocol adjacencies staying up and (ii) all routing protocol messages being received.



(a) Pre-config, Export RIB. (b) Migrate link and TCP. (c) Import RIB. (d) After migration.

Figure 4: Router grafting mechanisms – migrating a session with Router A (not shown) from router Migrate-from to router Migrate-to. The boxes marked bgpd and network stack are the software programs. The boxes marked RIB_A, $config_A$, and TCP_A are the routing, configuration, and TCP state respectively.

end-point. In addition, these statistics are vendor dependent and not well modularized in the router software implementations. As such, we do not migrate these statistics and instead allow the migrate-to router to initialize its own statistics as if it were establishing a new session.

3.3 Migrating TCP Connection & IP Link

As part of BGP session grafting, the TCP connection must move from the migrate-from router to the migrate-to router. Because we do not assume any support from the remote end-point, the migrate-to router must use the same IP addresses and sequence and acknowledgment numbers that the migrate-from router was using. In BGP, IP addresses are used to uniquely identify the BGP session end-points and not the router as a whole. Further, we assume the link between the remote end-point and the migrate-from (or migrate-to) router is a single hop IP network where the IP address is not used for reachability, but only for identification. As such, the session end-point can easily retain its address (and sequence and acknowledgment numbers) when it moves. That is, the single IP address identifying the migrating session can be disassociated from the migrate-from router and associated with the migrate-to router. Our architecture simply migrates the local state associated with the TCP connection from one router to another.

As with any TCP migration technique, the network must endure a brief period of time when neither router is responsible for the TCP connection. TCP has its own retransmission mechanism that ensures that the remote end-point retransmits any unacknowledged data. As long as the transient outage is short, the TCP connection (and, hence, the BGP session) remains up. TCP implementations tolerate a period of at least 100 seconds [10] without receiving an acknowledgment—significantly longer than the migration times we anticipate. The amount of TCP state is relatively small, and the two routers are close to one another, leading to extremely fast TCP migration times.

The underlying link should be migrated (e.g., by changing the path in the underlying programmable transport network) close to the same time as the TCP connection state, to minimize the transient disruption in connectivity. Still, the network may need to tolerate a brief period of inconsistency where (say) the TCP connection state has moved to the migrate-to router while the traffic still flows via the migrate-from router. During this period, we need to prevent the migrate-from router from erroneously responding to TCP packets with a TCP RST packet that resets the connection. This is easily prevented by configuring the migrate-from router's interface to drop TCP packets sent to the BGP port (i.e., 179). The migrate-from route *can* successfully deliver regular *data* traffic received during the transmission, as discussed later in Section 4.

3.4 Importing BGP Routing State

Once link and connection migration are complete, the migrate-to router can move its end-point of the BGP session from the 'inactive' state to the 'established' state. At this time, the migrate-to router can begin "importing" the RIBs received from the migrate-from router. However, the import process is not as simple as merely loading the RIB entries into its own internal data structures. The migrate-from and migrate-to routers could easily have a different view of the "best" route for each destination prefix, as illustrated in Figure 5. In this scenario, before the migration, A reaches E's prefixes over the direct link between them, and B reaches E's prefixes via A; after the migration, A should reach E's prefixes via B, and B should reach E's prefixes over the direct link. Similarly, suppose routers C and D connect to a common

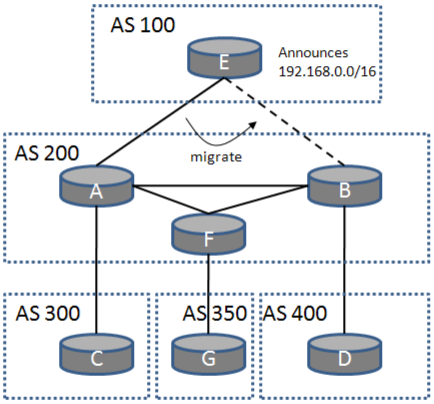


Figure 5: A topology where AS 200 has migrate-from router A, migrate-to router B, internal router F, and external routers C, D, and G, and remote end-point E.

prefix. Before the migration, E follows the AS path "100 200 300" (through C) to reach that prefix; after the migration E follows the AS path "100 200 400" (through D). Reaching these conclusions requires routers A and B to rerun the BGP decision process based on the new routes, and disseminate any routing changes to neighboring routers.

To make the process transparent to the remote end-point, we essentially emulate starting up a new session at router B, with router A temporarily playing the role of the remote end-point to announce the routes learned from E. This requires router A to replay the Adj-RIB-in state associated with E to router B. Router B stores these routes and reruns its BGP decision process, as necessary, to compute the new best routes to prefixes E is announcing. This will cause update messages to be sent to other routers within the AS and, sometimes, to external routers (like C and D). If the attributes of the route (e.g., the AS-PATH) do not change, as is the case in Figure 5, other ASes like AS 300 and AS 400 do not receive *any* BGP update message (since, from their point of view, the route has not changed), thus minimizing the overhead that router grafting imposes on the global BGP routing system.

Next, we update E with the best routes selected by B. Here, we take advantage of the fact that E has already learned routes from the migrate-from router A. The change in topology might change some of those routes, and we need to account for that. To do so, the migrate-to router runs the BGP decision process to compare its currently-selected best route to the route learned from the migrate-from router. If the best route changes, B sends an update message to its neighbors, including router E. This is in fact exactly the same operation the router would perform upon receiving a route update from any of its neighbors. We expect that routers A and B would typically have the same best route for most prefixes, especially if A and B are relatively close to each other in the IGP topology. As such, most of the time router B would not change its best route and hence would not need to send an update message to router E.

4 Correct Routing and Forwarding

Router grafting cannot be allowed to compromise the correct functioning of the network. In this section, we discuss how grafting preserves correct routing state (in the control plane) and correct packet forwarding (in the data plane), even when unexpected routing changes occur in the middle of the grafting process.

4.1 Control Plane: BGP Routing State

Routing changes can, and do, happen at any time. BGP routers easily receive millions of update messages a day, and these could arrive at any time during the grafting process – while the migrate-from router dumps its routing state, while the TCP connection and underlying link are migrated, or while the migrate-to router imports the routing state and updates its routing decisions. Our grafting solution can correctly handle BGP messages sent at any of these times.

While the migrate-from router dumps the BGP routing state: The goal is to have the in-memory Routing Information Base (RIB) be consistent with the RIB that was dumped as part of migration. Here, we take advantage of the fact that the dumping process and the BGP protocol work on a per-prefix basis. Consider a Adj-RIB-in with three routes (p1, p2, p3) corresponding to three prefixes, of which (p1 and p2) have been dumped already. When an update p3' (for the same prefix as p3) is received, the in-memory RIB can be updated since it corresponds to a prefix that has not been dumped, – to prevent dumping a prefix while it is being updated, the single entry in the RIB needs to be locked. If we receive an update p1' (for the same prefix as p1), processing it and updating the in-memory RIB without updating the dumped image will cause the two to be inconsistent – delaying processing the update is an option, but that would delay convergence as well. To solve this, we capitalize on BGP being an incremental protocol where any new update message implicitly withdraws the old one. Since we treat the dumped RIB as a sequence of update messages, we can process the update immediately and append p1' to the end of the dumped RIB to keep it consistent.

While the TCP connection and link are migrating: BGP update messages may be sent while the TCP connection and the underlying link are migrating. If a message is sent by the remote end-point, the message is not delivered and is correctly retransmitted after the link and TCP connection come up at the migrate-to router. If an update message is sent by another router to the migrate-from router over a different BGP session, there is not a problem because the migrate-from router is no longer responsible for the recently-rehomed BGP session. Therefore, the migrate-from router can safely continue to receive, select, and send routes. If an update message is sent by another router to the migrate-to router over a different BGP session, the migrate-to router can install the route in its Adj-RIB-in for that session and, if needed, update its selection of the best route – similar to when a route is received before the migration process.

While the migrate-to router imports the routing state: The final case to consider is when the migrate-to router receives a BGP update message while importing the routing state for the rehomed session. Whether from the remote end-point or another router, if the route is for a prefix that was already imported, there is no problem since the migration of that prefix is complete. If it is for a prefix that has not already been imported, only messages from the remote end-point router need special care. (BGP is an asynchronous protocol that does not depend on the relative order of processing for messages learned from different neighbors.) A message from the remote end-point must be processed after the imported route but we would like to process it immediately. Since the update implicitly withdraws the previous announcement (which is in the dump image), we mark the RIB entry to indicate that it is more recent than the dump image. This way, we can skip importing any entries in the dump image which have a more recent RIB update.

4.2 Data Plane: Packet Forwarding

Thus far, this paper has focused on the operation of the BGP control plane. However, the control plane's only real purpose is to select paths for forwarding data packets. Fortunately, grafting has relatively little data-plane impact. When moving a BGP session between blades in the same router, the underlying link does not move and the "best" routes do not change. As such, the forwarding table does not change, and data packets travel as they did before grafting took place – the data traffic continues to flow uninterrupted.

The situation is more challenging when grafting a BGP session from one router to another, where these two routers do not have the same BGP routing information and do not necessarily make the same decisions. Because the TCP connection and link are migrated *before* the migrate-to router imports the routing state, the remote end-point briefly forwards packets through the migrate-*to* router based on BGP routes learned from the migrate-*from* router. Since BGP route dissemination within the AS (typically implemented using iBGP) ensures that each router learns at least one route for each destination prefix, the two routers will learn routes for the same set of destinations. Therefore, the undesirable situation where the remote end-point forwards packets that the migrate-to router cannot handle will not occur.

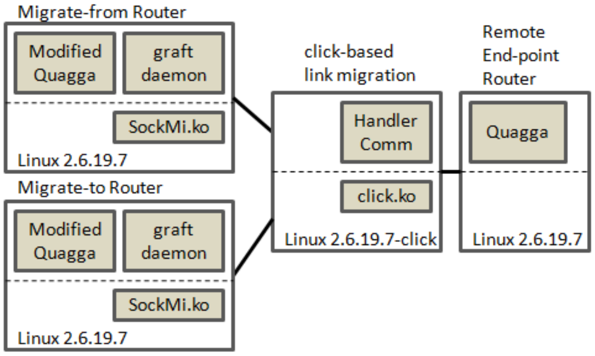


Figure 6: The router grafting prototype system.

Although data packets are forwarded correctly, the end-to-end forwarding path may temporarily differ from the control-plane messages. For example, in Figure 5, data packets sent by E will start traversing the path through AS 400, while E's control plane still thinks the AS path goes through AS 300. These kinds of temporary inconsistencies are a normal occurrence during the BGP route-convergence process, and do not disrupt the flow of traffic. Once the migrate-to router finishes importing the routes, the remote end-point will learn the new best route and control- and data-plane paths will agree again.

Correct handling of data traffic must also consider the packets routed *toward* the remote end-point. During the grafting process, routers throughout the AS forward these packets to the migrate-from router until they learn about the routing change (i.e., the new egress point for reaching these destinations). Since the migrate-from router knows where the link, TCP connection, and BGP session have moved, it can direct packets in flight there through temporary tunnels established between the migrate-from router and the migrate-to router.

5 BGP Grafting Prototype

We have developed an initial prototype to demonstrate router grafting. Figure 6 depicts the main components of the prototype. These include (i) a modified Quagga [11] routing software, (ii) the graft daemon for controlling the entire process, (iii) the SockMi [12] kernel module for TCP migration, and (iv) a Click [13] based data plane for implementing link migration.

The controlling entity in the prototype is the graft daemon. This is the entity that initiates the BGP session grafting, interacting with each of the other components to perform the necessary steps. We assume each graft daemon can be reached by an IP address. With this, the graft daemon on the migrate-from router will initiate a TCP connection with the daemon on the migrate-to router. Once established, the migration process follows the six general steps discussed in the following subsections.

5.1 Configuring the Migrate-To Router

In our architecture, configuration state is gleaned from a dump of the migrate-from router's configuration file, rather than its internal data structures. The graft daemon first extracts BGP session configuration from the configuration file of the migrate-from router, including the rules for filtering and modifying route announcements. Then the extracted configuration commands are applied to the migrate-to router. Our current implementation includes a simplistic parser for Quagga's commands for configuring BGP sessions[3]. In order to configure the migrate-to router before migrating the TCP connection, we added an 'inactive' state to the BGP state machine. We also added a configuration command to the Quagga command-line interface:

```
neighbor w.x.y.z inactive
```

that triggers the router to create all internal data structures for the session, without attempting to open or accept a socket with the remote end-point.

5.2 Exporting Migrate-From BGP State

Once the migrate-to router is configured, the grafting process can proceed to the second step, which is initiating the export of the routing state on the migrate-from router. The grafting daemon on the migrate-from router initiates the export process by calling a command in Quagga that we added:

```
neighbor w.x.y.z migrate out
```

When this command is executed, our modified Quagga software traverses the internal data structures, dumping the necessary routing state (Adj-RIB-in and the selected routes in the loc-RIB) to a file.

5.3 Exporting Migrate-From TCP State

Once the routing state is dumped, the modified Quagga calls the `export_socket` function as part of the SockMi API to migrate the TCP state. This function makes an `ioctl` call to the kernel module, passing the socket's file descriptor. The SockMi kernel module is a Linux kernel module for kernels 2.4 through 2.6—we tested with kernel version 2.6.19.7. The `ioctl` call causes the kernel module to interact with Linux's internal data structures. It removes the TCP connection from the kernel, writing the socket state to a character device. Note that part of this state is related to the protocol itself (e.g., the current sequence number) as well as the buffers (e.g., the receive queue and the transmit queue of packets sent, but not acknowledged). When this state is written, the kernel module sends a signal to the graft daemon on the migrate-from router, which can read from the character device and send to the daemon on the migrate-to router.

5.4 Importing the TCP State

The next step is to initiate the import of the TCP state at the migrate-to router. Upon receiving the state from the migrate-from router, the graft daemon on the migrate-to router first notifies Quagga that it is about to import state for a given 'inactive' session. This is done through a command we added:

```
neighbor w.x.y.z migrate in
```

Upon executing the command, our modified Quagga invokes the `import_socket` function in the SockMi API. This function blocks until a TCP connection is imported. During this time, the graft daemon makes an `ioctl` to the SockMi kernel module. The graft daemon then passes the TCP session state to a character device which is read by the kernel module. The SockMi kernel module accesses the Linux data structures to add a socket with that TCP connection state, which unblocks the `import_socket` function.

5.5 Migrating the Layer-Three Link

At this point, the graft daemon of the migrate-to router triggers the migration of the underlying link. This includes removing the migrating session's IP address from the migrate-from router, adding the IP address to the migrate-to router, and migrating the layer-two link. As we did not have access to equipment to use a programmable transport network, we instead built our own simple layer-two network that connects both the migrate-from and migrate-to router to the remote end-point with a Click [13] configuration that emulates a 'programmable

[3] As we add support for XORP, we will develop a more complete parser as the configuration will require translating between configuration languages—generally a hard problem, though easier in our case because we focus on a relatively narrow aspect of the configuration.

transport'. This Click configuration performs a simple switching primitive that connects the remote end-point to either the migrate-from or the migrate-to router. In one setting, packets from the migrate-from router are sent to the remote end-point router, packets from the migrate-to router are dropped, and packets from the remote end-point router are sent to the migrate-from router. With the alternative setting, the reverse occurs, forming a link between the migrate-to router and the remote-end point router. This switch value is settable via a handler, making it accessible to the graft daemon running on the migrate-from router.

5.6 Importing Routing State

As the final step, when the importing of the TCP connection is complete and the `import_socket` function is unblocked, the modified Quagga reads the routing state, which was stored in a file when the local graft daemon read it in from the graft daemon running on the migrate-from router. Much as the "normal" operation of the router, which receives a BGP message from a socket and then calls a function to handle the update, the importing process will read the Adj-RIB-in from a file and call the same function to process the routing update. For comparing the RIB from the migrate-from router to the migrate-to router, the importing process reads the route from the file, looks up the route in the local RIB, and compares them. If they differ, it will use existing functions to send out the route to the peer.

6 Optimizations for Reducing Impact

Grafting a BGP session requires incrementally updating the remote end-point as well as the other routers in the AS. In this section, we present optimizations that can further reduce the traffic and processing load imposed on routers not directly involved in the grafting process. These optimizations capitalize on the knowledge that grafting is taking place and the routers' local copy of the routes previously learned from the remote end-point. First, we discuss how we can keep routers from sending unnecessary updates to their eBGP neighbors. Second, we then discuss how the majority of iBGP messages can be eliminated. Finally, we consider the intra-cluster router case where the routes do not change.

6.1 Reducing Impact on eBGP Sessions

Importing routes on the migrate-to router, and withdrawing routes on the migrate-from router, may trigger a flurry of update messages to other BGP neighbors. Consider the example in Figure 5, where before grafting router E had announced 192.168.0.0/16 to router A, which in turn announced the route to B and C. Eventually two things will happen: (i) the migrate-from router A will *remove* the 192.168.0.0/16 route from E and (ii) the migrate-to router B will *add* the 192.168.0.0/16 route from E. Without any special coordination, these two events could happen in either order.

If A removes the route before B imports it, then A's eBGP neighbors (like router C) may receive a withdrawal message, or briefly learn a different best route (should A have other candidate routes), only to have A reannounce the route upon (re)learning it from B. Alternatively, if B adds the route before A sends the withdrawal message to C, then A may have both a withdrawal message and the subsequent (re)announcement queued to send to router C, perhaps leading to redundant BGP messages. In the first case, C may temporarily have no route at all, and in the second case C may receive redundant messages. In both cases these effects are temporary, but we would like to avoid them if possible.

To do so, rather than deleting the route, A can mark the route as "exported"—safe in the knowledge that, if this route should remain the best route, A will soon (re)learn it from the migrate-to router B. For example, suppose the route from E is the *only* route for the destination prefix—then A would certainly (re)learn the route from B, and could forgo withdrawing and reannouncing the route to its other neighbors. Of course, if A does not receive the announcement (either after some period of time or implicitly through receiving an update with a different route for that prefix), then it can proceed with deleting the exported route.

So far we only considered the eBGP messages the migrate-from router would send. A similar situation can occur on the eBGP sessions of the other routers in the AS (e.g., router F). This is because these other routers must be notified (via iBGP) to no longer go through A for the routes learned over the migrating session (i.e., with E). Therefore, the migrate-from router must send out withdrawal messages to its iBGP neighbors and the migrate-to router must send out announcements to its iBGP neighbors. This may result in the other routers in the AS (e.g., router F) temporarily withdrawing a route, temporarily sending a different best route, or sending a redundant update to their eBGP neighbors. Because of this, we have the migrate-from router send the marked list to each of its iBGP neighbors and a notification that these all migrated to the migrate-to router – this list is simply the list of prefixes, not the associated attributes. We expect this list to be relatively small in terms of total bytes. With this list, the other routers in the AS can perform the same procedure, and eliminate any unnecessary external messages.

6.2 Reducing Impact on iBGP Sessions

While using iBGP unmodified is sufficient for dealing with the change in topology brought about by migration, it is still desirable to reduce the impact migration has on the iBGP sessions. Here, since the route-selection policy will likely be consistent throughout an ISP's network, we can reduce the number of update messages sent by extending iBGP (an easier task than modifying eBGP). When the migrate-from and migrate-to routers select the same routes, the act of migration will not change the decision. Since all routers are informed of the migration, the iBGP updates can be suppressed (the migrate-from router withdrawing the route and the migrate-to router announcing the route). When the migrate-from and migrate-to routers select different routes, it is most likely due to differences in IGP distances. For the migrate-to router, the act of migration will cause all routes learned from the remote end-point router to become directly learned routes, as opposed to some distance away, and therefore the migrate-to router will now prefer those routes (except when the migrate-to router's currently selected route is also directly learned). This change in route selection causes the migrate-to router to send updates to its iBGP neighbors notifying them of the change. However, since it is more common to change routes, we can reduce the number of updates that need to be sent with a modification to iBGP where updates are sent when the migrate-to router keeps a route instead of when it changes a route. Other routers will be notified of the migration and will assume the routes being migrated will be selected unless told otherwise.

6.3 Eliminating Processing Entirely

Re-running the route-selection processes is essential as migration can change the topology, and therefore change the best route. When migrating within a cluster router, the topology does not change, and therefore we should be able to eliminate processing entirely. The selected best route will be a consistent selection on every blade. Therefore, even when migrating, while the internal data structures might need to be adjusted, no decision process needs to be run and no external messages need to be sent. In fact, there is no need for any internal messages to be sent either. With the modified iBGP used for communication between route processor blades, the next hop field is the next router, not the next processor blade – i.e., iBGP messages are only used to exchange routes learned externally and do not affect how packets are forwarded internally. Therefore, upon migration, there is no need to send an update as the routes learned externally have already been exchanged.

While exchanging messages and running the decision process can be eliminated, transferring the routing state from the exporting blade to the importing blade is still needed. Being the blade responsible for a particular BGP session requires that the local RIB have all of the routes learned over that session. While some may have been previously announced by the migrate-from blade, not all of them were. Therefore, we need to send over the Adj-RIB-in for the migrating session in order to know all routes learned over that session as well as which subset of routes the migrate-from blade announced were associated with that session.

7 Performance Evaluation

In this section, we evaluate router grafting through experiments with our prototype system and realistic traces of BGP update messages. We focus primarily on control-plane overhead, since data-plane performance depends primarily on the latency for link migration—where our solution simply leverages recent innovations in programmable transport networks. First, we evaluate our prototype implementation from Section 5 to measure the grafting time and CPU utilization on the migrate-from and migrate-to routers. Then we evaluate the effectiveness of our optimizations from Section 6 in reducing the number of update messages received by other routers.

7.1 Grafting Delay and Overhead

The first experiment measures the impact of BGP session grafting on the migrate-from and migrate-to routers. To do this we supplemented the topology shown in Figure 5 with a router adjacent to E (in a different AS) and a router adjacent to B (in a different AS). These two extra routers were fed a BGP update message trace taken from RouteViews [14]. This essentially fills the RIB of B and E with routes that have the same set of prefixes, but different paths. We used Emulab [15] to run the experiment on servers with 3GHz processors and 2GB RAM.[4]

The time it takes to complete the migration process is a function of the size of the routing table. The larger it is, the larger the state that needs to be transferred and the more routes that need to be compared. To capture this relationship, we varied the RIB size by replaying multiple traces. The results, shown in Figure 7, include both the case where migration occurs between routers (when the migrate-to router must run the BGP decision process) and the case where migration is between blades (where the decision process does not need to run because the underlying topology is not changed). The "between blades" curve, then, illustrates the time required to transfer the BGP routes and import them into the internal data

[4] This is roughly comparable to the route processors used in commercially available high-end routers.

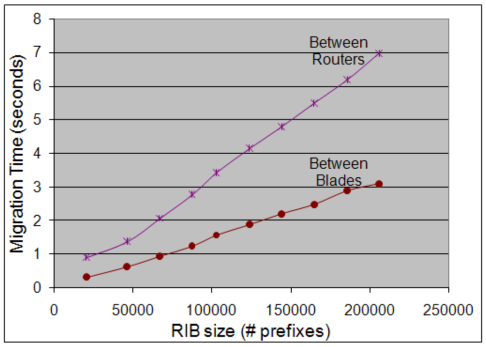


Figure 7: BGP session grafting time vs. RIB size.

structures. Note that these results do not imply that TCP needs to be able to handle this long of an outage where packets go unacknowledged – the TCP migration process takes less than a millisecond. Instead, when compared to rehoming a customer today, where there is downtime measured in minutes, the migration time is small. In fact, since in our setup AS100 and AS200 have a peering agreement, the actual migration time would be less if AS100 were a customer of AS200 (since AS100 would announce fewer routes to AS200).

The CPU utilization during the grafting process is also important. The BGP process on the migrate-from router experienced only a negligible increase in CPU utilization for dumping the BGP RIBs. The migrate-to router needs to import the routing entries and compare routing tables. For each prefix in the received routing information, the migrate-to router must perform a lookup to find the routing table entry for that prefix. Figure 8 shows the CPU utilization at 0.2 second intervals, as reported by *top*, for the case where the RIB consists of 200,000 prefixes. There are three things to note. First, the CPU utilization is roughly constant. This is perhaps due to the implementation where the data is received, placed in a file, then iteratively read from the file and processed before reading the next. This keeps the CPU utilization at only a fraction as computation is mixed with reads from disk. Second, the CPU utilization is the same for both migrating between routers and migrating between blades. The case between routers merely takes longer because of the additional work involved in running the BGP decision process. Third, migration can be run as a lower priority task and use less CPU but take longer – preventing the migration from effecting the performance of the router during spikes in routing updates, which commonly results in intense CPU usage during the spikes.

7.2 Optimizations for Reducing Impact

While the impact on the migrate-from and migrate-to routers is important, perhaps a more important metric is

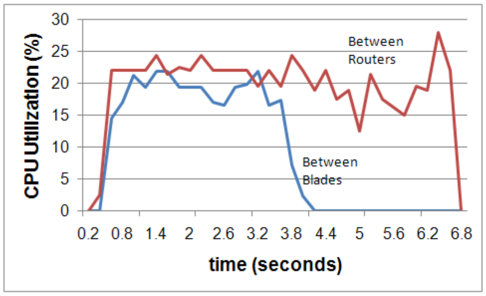
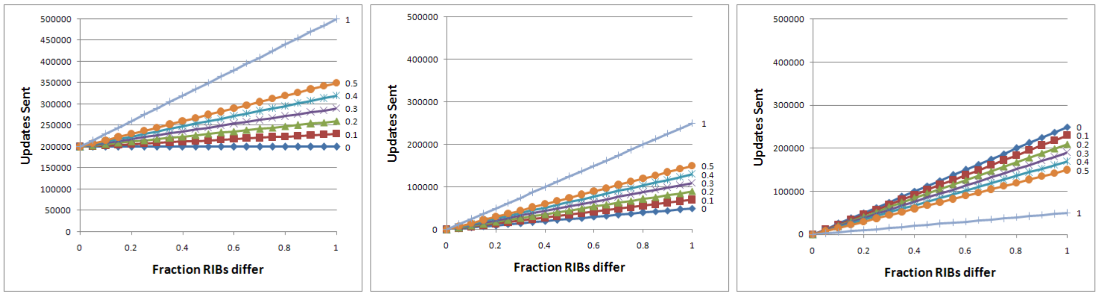


Figure 8: The CPU utilization at the migrate-to router during migration, with a 200k prefix RIB.

the impact on the routers *not* involved in the migration, including other routers within the same AS as well as the eBGP neighbors. If the overhead of grafting is relatively contained, network operators could more freely apply the technique to simplify network-management tasks.

First and foremost, the remote end-point experiences an overhead directly proportional to the number of additional BGP update messages it receives. The number of messages depends on how many best routes differ between the migrate-from and migrate-to router—the migrate-from router must send an update message for every route that differs. The exact amount depends heavily on the proximity of the migrate-from and migrate-to routers—if the two routers are in the same Point-of-Presence of the ISP, perhaps no routes would change. As such, we do not expect this overhead to be significant. Since the sources of overhead for the remote end-point are relatively well understood, and it is difficult to acquire the kinds of intra-ISP measurement data necessary to quantify the number of route changes, we do not present a plot for this case.

Perhaps the more significant impact is on the other routers, both within the AS and in other ASes, that may have to learn new routes for the prefixes announced by the remote end-point. To evaluate this, we measured the number of updates that would be sent as a function of the fraction of prefixes where the migrate-from router had selected a different route than the migrate-to router. By doing so, this covers the entire range of migration targets (i.e. it does not limit our evaluation to migration within a PoP). Recall that this difference is what needs to be corrected. Also recall that the prefixes being considered here are the ones learned from the router at the remote end-point of the session being migrated, not the entire routing table, as these are the routes that could impact what is sent to other routers. For our measurement, we use a fixed set of 100,000 prefixes. However, the results are directly proportional to the number of prefixes, and can therefore be scaled appropriately – for migrating a customer link, the number of prefixes would be significantly smaller, for migrating a peering link, the number



(a) Without optimization. (b) Reducing eBGP impact. (c) Reducing iBGP impact.

Figure 9: Updates sent as a result of migration.

of prefixes could be higher.

The results are shown in Figure 9, with the three graphs representing the three different cases as discussed in Section 6: (a) direct approach with no optimizations, (b) optimizations to reduce eBGP messages by capitalizing on redundant information in the network, and (c) optimizations to reduce iBGP messages by treating the route selection changing as the common case. For the graphs, each line represents a fixed fraction of differing routes that change the selected route as a result of the grafting. For example, consider where the migrate-from router selects a particular route different than the migrate-to router. In this case, after migration, the migrate-to router selects the route the migrate-from selected (i.e., it changes its own route). Each line represents the fraction of times this change occurs—for example, the line labeled 0.2 in Figure 9 is where 20% of the routes that differ will change to the routes selected by the migrate-from router.

There are several things of note from the graphs. First is that the direct (unoptimized) approach must send significantly more messages. In the case where the selected routes do not differ much, which we consider will be a most likely scenario, the optimized approaches hardly send any messages at all. Second, when comparing Figure 9(b) with Figure 9(c), we can see that depending on what would be considered the common case, we can choose a method that would result in the fewest updates. For (b), the assumption is that when the routes differ, the migrate-to router will not change to the routes the migrate-from selected. Whereas in (c), the assumption is that when the routes differ, the migrate-from router will change to the routes the migrate-from router selected. The reason they would change is that the routes learned from the remote end-point of the session being migrated will now be directly learned routes, rather than via iBGP. It is likely that the policy of route selection is consistent throughout the ISPs network, and therefore differences will be due to IGP distances and changing the router will change those routes to be more preferable. We are working on characterizing when these differences would occur in order to enable us to predict the impact a given migration might have. Third, and perhaps most important, migration can be performed with minimal disruption to other routers in the likely scenario where there are few differences in routes selected.

8 Related Work

High availability and ease of network management are goals of many systems, and therefore router grafting has much in common with them. In particular, ones that attempt to minimize disruptions during planned maintenance. One possibility is to reconfigure the routing protocols such that traffic will no longer be sent to the router about to undergo maintenance [16, 17]. Alternatively, others have decoupled the control plane and data plane such that the router can continue to forward packets while the control plane goes off-line (e.g., rebooted) [18, 19]. However, unlike router grafting, these require modifications to the remote end-point router and they are only useful for temporarily shutting down the session on a given physical router, rather than enabling the session to come back up on a different router as in router grafting.

In this regard, router grafting shares more in common with VROOM [3], which makes use of virtual machine migration [20] to ease network management. Maintenance could be performed without taking down the router simply by migrating the virtual router to another physical router. This requires the two physical routers to be compatible (running the same virtualization technology), a limitation router grafting does not have. In fact, router grafting does not rely on virtual machine technology. Kozuch showed the ability to migrate without the use of virtualization [21], but did so at the granularity of the entire operating system and all running processes. With a coarse granularity, the physical router where the virtual router is being migrated to must be able to handle the entire virtual router's load.

Router grafting is also similar to the RouterFarm work [6], which targeted re-homing a customer. However, it required restarting the session and is more disruptive than router grafting. Along similar lines, high-availability routers enable switching over to a different router or blade in a router [22]. This, however, is done either through periodically check-pointing, which preserves the memory image, or running two complete instances of the router software concurrently, which is an inefficient use of resources.

While we presented router grafting in the context of a BGP session, we envision it being more general. Along these lines, partitioning the prefix space across multiple routers or blades is a possibility.ViAggre [23] partitions the prefix space across multiple routers, however it is a static architecture not one which dynamically repartitions the prefix space as router grafting could.

Finally, we made use of TCP socket migration to handle change or disruption in end-points. One alternative is to modify the TCP protocol to include the ability to change IP addresses [24]. Since the IP address of the end-points in router grafting can remain the same, we do not need this capability, but could make use of it.

9 Conclusions

Router grafting is a new technique that opens many new possibilities for managing a network. It does this by enabling, without disruption, the migration of a routing session between (i) physical routers, (ii) blades in a cluster router, and (iii) routers from different vendors. We were able to do this while being transparent to the remote end-point. We handled the changes in topology through incremental updates, only sending out the necessary updates to convey the difference. Importantly, we did not affect the correctness of the network as the data plane will continue to forward packets and routing updates do not cause the migration to be aborted.

Going forward, we plan to explore the motivating applications for router grafting to further demonstrate the usefulness of our new technique. We are particularly interested in exploring the role of router grafting in traffic engineering. Finally, this work raises interesting questions about what exactly a router is, and the various ways routers can be "sliced and diced." We plan to explore these questions in our ongoing work.

References

[1] S. Agarwal, C. Chuah, S. Bhattacharyya, and C. Diot, "Impact of BGP dynamics on router CPU utilization," in *Passive and Active Measurement*, April 2004.

[2] J. Gottlieb, A. Greenberg, J. Rexford, and J. Wang, "Automated provisioning of BGP customers," *IEEE Network Magazine*, November/December 2003.

[3] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual Routers on the Move: Live Router Migration as a Network-Management Primitive," in *ACM SIGCOMM*, August 2008.

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

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

[6] 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.

[7] 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.

[8] M. Tahir, M. Ghattas, D. Birhanu, and S. N. Nawaz, *Cisco IOS XR Fundamentals*. Cisco Press, 2009.

[9] "IETF draft: MRT routing information export format," July 2009. `http://tools.ietf.org/id/draft-ietf-grow-mrt-10.txt`.

[10] R. Braden, "Requirements for Internet Hosts - Communication Layers." RFC 1122, October 1989.

[11] "Quagga software routing suite," `www.quagga.net`.

[12] M. Bernaschi, F. Casadei, and P. Tassotti, "SockMi: a solution for migrating TCP/IP connections," in *Proc. Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, 2007.

[13] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek, "The Click modular router," in *ACM Trans. Comp. Sys.*, August 2000.

[14] "Route views project," `http://www.routeviews.org`.

[15] B. White, J. Lepreau, L. Stoller, R. Ricci, G. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," in *OSDI*, December 2002.

[16] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, "Dynamics of hot-potato routing in IP networks," *IEEE/ACM Trans. Networking*, December 2008.

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

[18] A. Shaikh, R. Dube, and A. Varma, "Avoiding instability during graceful shutdown of multiple OSPF routers," *IEEE/ACM Trans. Networking*, vol. 14, pp. 532–542, June 2006.

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

[20] 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. Networked Systems Design and Implementation*, May 2005.

[21] M. A. Kozuch, M. Kaminsky, and M. P. Ryan, "Migration without virtualization," in *Proc. Workshop on Hot Topics in Operating Systems*, May 2009.

[22] "Cisco IOS high availability curbs downtime with faster reloads and upgrades." `http://www.cisco.com/en/US/products/ps6550/prod_white_papers_list.html`.

[23] H. Ballani, P. Francis, T. Cao, and J. Wang, "Making Routers Last Longer with ViAggre," in *Proc. of USENIX Symposium on Networked Systems Design and Implementation*, April 2009.

[24] A. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility," in *Proc. ACM MOBICOM*, (Boston, MA), August 2000.