The "Platform as a Service" Model for Networking

Eric Keller
Princeton University

Jennifer Rexford
Princeton University

Abstract

Decoupling infrastructure management from service management can lead to innovation, new business models, and a reduction in the complexity of running services. It is happening in the world of computing, and is poised to happen in networking. While many have considered this in the context of network virtualization, they all focus on one model – overlaying a virtual network of multiple virtual routers on top of a shared physical infrastructure, each completely isolated from the others through the use of virtualization. In this paper we argue for a different approach, where those running the service are presented with the abstraction of a *single* router in order to enable them to focus solely on their service rather than worrying about managing a virtual network as well. We discuss the abstraction of a single router, and the challenges of mapping the collection of abstract routers (from different parties) to the distributed and shared physical infrastructure.

1 Introduction

In the world of computing, a shift has begun towards the use of infrastructures which are hosted and shared (i.e., cloud computing). This has increased the level of innovation by enabling companies to come out with new web services for less cost, created new business models where a party can lease out slices of servers on demand with a pay-per-use model, and even simplified management in private (non-hosted) networks by enabling a company to more easily run independent services on its own servers. We believe that the same will be true of networking where many applications would greatly benefit from 'in-network' functionality beyond the basic connectivity offering of today.

This is already underway, as many researchers have been exploring this model for years – new business models decoupling who owns the physical infrastructure (infrastructure provider) from who runs and configures the service (service provider) [1, 2, 3], shared testbeds using virtualization to enable innovation in networking research [4, 5], and capitalizing on virtualization to ease the burden of running multiple services in a single ISP [6, 7]. However, these studies have focused only on a single model – that of a network of virtual routers, where one can lease or use a slice of physical routers connected together via partitioned links. Taking from the terminology used in cloud computing, we characterize this as analogous to the infrastructure as a service (IaaS) model since it simply slices the physical resources.

However, in the virtual networks model, the virtual network needs to be managed the same way the physical network is – e.g., dealing with failure and link congestion. In this paper we explore an alternative model – the platform as a service (PaaS) model. Instead of exposing the underlying physical network and topology, we propose the abstraction presented to customers should be a single router. As routers today include both route processing blades as well as general compute blades, a trend that will likely continue, the single router abstraction is general enough to cover all in-network functionality (i.e., not just routing). This model benefits both the customers, as they can run their service without needing to manage a physical network, and the infrastructure providers, as they can use their platform as a source of differentiation.

In the remainder of the paper we first discuss how applications would benefit from more in-network functionality than what is offered by ISPs today (Section 2). We then discuss why virtual networks place too much burden on the customer (Section 3). Next, our proposed router platform model is presented where we discuss both the abstraction and the challenges of realizing it on a distributed and shared network (Section 4). We follow that with a discussion on how the router platform model can help ease private network management (Section 5) and conclude (Section 6).

2 Basic Connectivity is Not Enough

What ISPs offer today is basic connectivity – i.e., the ability to communicate with any other IP address at a given bandwidth. However, as many have argued, this is not enough as many applications can benefit from 'in-network' functionality. Here, we briefly discuss a few of these (this list is by no means exhaustive).

Customer controlled routing: Despite the large path diversity available in ISPs, the mechanisms of the Border Gateway Protocol (BGP), used for inter-domain routing, are such that a single route must be chosen at each router and each route reflector. This means that this diversity is not made available to customers, instead forcing the ISP to announce the same route to each customer. At the same time, each customer is different and has its own preferences of what constitutes a route as being the best. One may desire a low latency path, another may desire a path that does not go through certain countries, while a third may desire a low cost path. If the ISP went beyond basic connectivity, they could in fact enable different neighbors to receive different routes by giving the customer control over route selection (either as coarse-grained selection, such as "prefer lowest latency," or by giving complete control to the customer) [8] [9].

Cloud computing: Cloud computing is an exploding business where enterprises and providers of web services lease compute resources from a cloud provider. However, the customer is simply given servers and the capability to reach each server (both externally and internally). The customer does not have any ability to configure the network components (routers and switches), as they would in their own private facilities [10]. This limits the middle-box type functionality (such as a firewall or intrusion prevention system) that enterprise networks commonly have. It also hinders the ability to use virtual private networks (VPNs), which enterprises commonly use for inter-site communication, as existing cloud providers do not provide enough capability to enable the customers to securely attach to a VPN endpoint [11].

Gaming and Live video streaming: Online games, whether action oriented first-person shooters or more exploratory virtual worlds, have a great need for the ability to efficiently distribute updates to all of the players. Live video streaming also requires an efficient distribution of the stream. Publish/subscribe and multicast are ideal for both of these applications. These applications also share the need for on-path processing (e.g., to aggregate game updates or to transcode the video stream). However, an efficient distribution or on-path processing are not part of the basic connectivity offering.

Network monitoring: Many applications have the need to monitor the condition of the network (e.g., the availability, congestion, or latency of paths) in order to troubleshoot or perform, for example, server selection. This includes a number of techniques such as observing BGP updates or performing ping tests. Given the distributed nature of the Internet, monitoring servers must be setup throughout the world. For this reason, in-network visibility is a more accurate and efficient way to monitor the condition of the network than those limited by the basic connectivity offering.

3 Virtual Networks Are Too Much

To go beyond basic connectivity, ISPs can add functionality to their network and offer it to their customers. However, they are limited to the geographic foot print of their network and innovation is limited to a single party (the ISP). Overlay networks have arisen as a way around this, enabling 3rd parties to run specialized network functionality (e.g., content delivery networks) by setting up servers throughout the world and using the Internet as 'links' between the servers. Still, each of these are one off deployments and therefore researchers have proposed virtual networks as a generalization which enables multiple parties to create custom networks on a shared physical network substrate [1, 2, 3, 4, 5].

In the virtual networks model, multiple virtual networks can co-exist on a shared physical substrate through the use of virtualization. A virtual network is a collection of virtual routers connected in a topology via virtual links. A virtual router is a 'slice' of a physical router (i.e., the physical router's resources are partitioned and each virtual router is isolated from one another). Virtual links are created through partitioning the physical link's bandwidth and creating tunnels when adjacent virtual routers are not located on adjacent physical routers. Those wanting to create a custom network (either for themselves or to sell as a service to others) can do so by leasing resources from an infrastructure provider (the party that owns and maintains the physical network).

However, while overcoming the limitations of basic connectivity, there are problems with the virtual networks model. These problems are centered around the fact that managing a virtual network is similar to managing a physical network. There is limited bandwidth on the virtual links between virtual routers, and so the customer must engage in traffic engineering to optimally use the virtual link bandwidth. Further, as there is a one to one mapping of virtual routers to physical routers, the failure of a physical component (line card, route processor blade, or link) is visible to the virtual networks that use those components. Because of this, the customer must be able to cope with failure (e.g., by providing redundancy). Making each customer manage the virtual network, with the same burden as managing a physical network, is unappealing to the customers, who really only want to run their service.

At the same time the virtual network model may not be desirable for the infrastructure provider either. First, when the same party that runs the services also owns the physical infrastructure, planned maintenance events can be coordinated (e.g., reconfigure the service to route around a router that is going to be shut down). This coordination is made more difficult when the two parties are different. Virtual router migration [7] is a possible way to deal with this, yet it still takes time and depending on what the customer's router is doing, may not be applicable. Second, partitioning is an inefficient use of resources. The infrastructure provider has no control over, for example, how much traffic each virtual router will send to each other virtual router, and therefore must allocate enough resources so the customers receive the service level they are promised. Third, and perhaps most important, the infrastructure provider is simply offering a commodity service – there is little opportunity to differentiate themselves from other infrastructure providers other than cost. This is unappealing to the infrastructure providers, and something the ISPs are grappling with in today's basic connectivity offering.

4 A Router Platform is Just Right

Virtual networks go too far in compensating for the limitations of the basic connectivity model. Instead, we propose presenting customers with a platform that is decoupled from the physical infrastructure – making it so the customers only have to manage their services and the infrastructure provider has freedom to manage the physical infrastructure. This platform will look very much like a router today, making it both familiar to network operators who are used to specifying routing policies in terms of router configuration, and generic enough to cover any type of in-network functionality. In the remainder of this section we discuss this single router abstraction and how it is realized on a distributed and shared physical infrastructure.

4.1 The Single Router Abstraction

What the customer really cares about is being able to configure its specific policies and how packets are handled. As such, we propose that they are provided the view of a single router. Important to note is that we view a router in a broader sense than what they are traditionally thought of. That is, routers are becoming more and more powerful and converging towards a large, heterogeneous distributed system (today's high end routers are already there). They have many line cards to perform data plane functions, route processing blades which runs routing software to calculate paths through the network, general purpose processing blades allowing general software to be run (control or packet handling), and a switching fabric that connects all of the cards together.

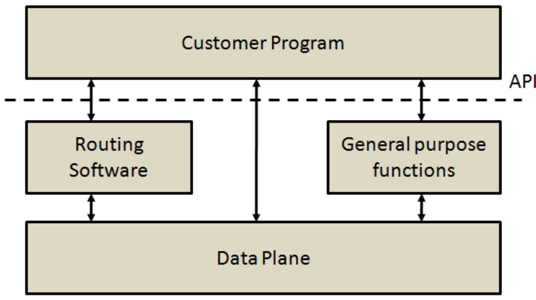

Figure 1: The single router abstraction.

From this, the platform we propose as the abstraction presented to the customer is as show in Figure 1, covering the three main processing components in a router – the route processing, the data plane, and the general purpose processing.

Interactive program : Rather than a static configuration file, we envision the programming environment being an application programming interface (API) – in which case the customer would provide an executable program (e.g., a Python script). This program would include an initialization routine, which would use each component's API to configure the router in its initial state. Then, the program can dynamically update this configuration. This may come as a result of receiving a message (e.g., from one of the customer's servers) indicating a change to be made. Or, it may come as a result of a network event – the definition of which is broad and will be specific to each component, but may include things like links going down or cost of using a link changing. To handle these events, the customer's program will provide callback functions and register itself as being interested in a particular event.

Routing : The routing component provides the ability to customize path selection. Much as with today's configuration mechanism of routers, customers will be able to specify sessions with neighboring routers. This will include both sessions to the customer's own routers (e.g., in its own facilities, or run by another infrastructure provider) as well as sessions to routers that the infrastructure provider has a business relationship with (either its upstream provider or a peer). Note that, specifying a session is for configuration only – sessions are shared among all customers. With this configuration, the customer can also specify policy which determines which routes are preferred, what modifications to make to routes, and which neighboring routers to announce updates to[1].

To be able to select routes, the customer needs to know information about the various links it has the option to choose among. First, it needs to be able to query what the available links are. Second, it needs to know certain metrics for each link – such as relative cost. Previously, the ISP would choose a given link (e.g., the cheapest) and that is what the customer would use. With this, they have the option of using a more expensive link if they desire. We envision that these metrics can and will change, and this can be notified to the user through the event mechanism, in which case the user can update the configuration.

Data plane : The configuration of today's routers includes configuring the data plane functionality in addition to the routing policies. We view them as separate functions and as such treat the data plane as a separate module from the routing. For this, we are not suggesting that the customer's program will interact directly with the line cards. Only that a sub-set of the capabilities of a line card will be exposed to the customer's program. This includes such functionality as being able to configure multicast groups or setting up access control lists (ACLs). As an extension to the current static configuration, the data plane can also have events that a customer's script can act upon. Events here are intended to be notifications about aspects the customer cares about (e.g., the traffic rate on a given link has reached a certain threshold) and not those that are handled by the routing software already (such as a link failing), though even those could be informative.

General-purpose processing : The general purpose processing module is the most generic, and therefore, difficult to generalize. However, it will be presented to the user as though they are configuring a special purpose machine – such as a server which performs measurement, or a firewall, or file caching storage layer, or any other functionality that benefits from being inside of the network. This is really a way for the infrastructure provider to differentiate itself from other infrastructure providers. Longer term, we believe that customers may be able to write their own software modules – including per-packet handling, per-flow handling, or control only processing. This would be facilitated by virtualization technologies which would enable isolation of the module from other modules, ensuring other customers are not affected by faults in the customer's software.

4.2 The Physical Infrastructure Reality

Our proposal is to provide each customer the ability to program a single router. There are two main challenges of this when realizing it on the actual physical infrastructure: (i) that the infrastructure is distributed and (ii) that the infrastructure is shared. Each of these is discussed below.

4.2.1 Distributed Router

The abstraction of a single router does not match the architecture of networks today consisting of many routers, which collectively handle the routing and forwarding workload, and a variety of special purpose machines. In fact, routers are becoming more like single box versions of the networks they run in. As such, we will generalize the definition of a physical router (as opposed to an abstract router) to be the combination of route processing blades, packet processing blades, and general-purpose processing blades, all connected via a large switching fabric. The switching fabric can either be a network of routers (e.g., running OSPF), a network of Ethernet switches, or a switch internal to the router (e.g., a Clos or crossbar).

This idea of distributing the workload among many routers and special purpose machines is not really anything new, operators readily do this today. However, due to the decoupling of who owns and maintains the physical infrastructure from who configures the router, the process must be highly automated and must be able to change. The need to be able to dynamically change comes from the fact that the infrastructure provider does not know the customer's exact workload or exactly which functionality will be used. It also simplifies workload distribution since if the placement turns out to be a poor selection, it can be changed.

Automatic distribution then involves four steps, each of which is briefly discussed below.

Choose a placement : There are a number of factors that go into deciding where to place a certain resource. First, there is the functionality requirement of the customer's router – if it is using some capability (e.g., multicast or a firewall), that processing must be done on a router that supports that capability. Second, the processing requirements of the individual parts of a customer's router must be met – we can partition routing sessions across multiple route processors and we can replicate middle-box functionality in order to scale. The rate at which routing updates (for routing software) or data packets (for general-purpose software) arrive along with the estimate of how much processing each request requires, will determine how much each processor can support. Third, basic performance factors such as latency and maximum throughput need to remain reasonable. Fourth, the traffic on the internal network must be taken into account. This may come from internal control communication (e.g., iBGP traffic), data traffic going from ingress to egress

[1]For multiple links to the same neighboring AS, when all else is equal, the customer can say it does not care and the infrastructure provider is free to choose (e.g., based on location).

router, and data traffic to each middle-box in succession.

Configure inter-processor communication : When a placement is decided upon, the inter-processor communication needs to be set up. Each middle-box type module will have it's own inter-processor communication (e.g., a caching solution might exchange what is currently being cached with one another). For routing, the inter-processor communication is an exchange of routes and is typically set up via the configuration language (e.g., specify all of the iBGP routing sessions) – since the customer's view is that of a single router, it does not need to configure the iBGP sessions, they are automatically configured by the infrastructure provider. Once configured, each processor typically selects a best route locally from the sessions terminated at that processor and sends those to each other so that each will select an overall best route[2].

Tune the 'switch' : The impact on the internal network is taken into account when deciding placement. However, since it it not the only factor, there is also the need to optimize the internal network through traffic engineering. This may, for example, include adjusting IGP routing protocol parameters, such as link weights.

Dynamically adapt : The placement process uses estimates to determine the best placement. This limits the effectiveness as there is limited information when performing the placement and the demands may change over time. Because of this, we view the process as a more dynamic one where the placement can change. This requires monitoring all of the key inputs (CPU usage, update frequency, traffic patterns, etc.). From this, we can re-run the placement algorithms. However, changing placement is not always easy. State may need to be transferred and the actual process may require some processing – e.g., migrating a BGP session involves exchanging the routing information and then re-running routing processes [12]. Therefore, some notion of the cost of migrating a particular workload is taken into account.

4.2.2 Shared Infrastructure

Going beyond supporting the single router abstraction on a distributed infrastructure, the model must also support multiple customers, each specifying their own abstract router. For this, the individual components of the distributed network need to themselves have support for supporting multiple customers.

The need to support multiple customers has been studied in the context of virtual networks. However, since in our model, each customer is only allowed to configure the router, as opposed to run their own routing processes, it is more efficient to simply run a single control process handling all customers' routing sessions. In fact, the abstract routers cannot be completely isolated as they can share a session with a neighboring router – if multiple customers specify the same neighboring router as a peer, only a single session is established (as that is what the neighboring router expects). To handle this, when a route is received, the routing software will replicate the message, tag each with a customer identifier, and process each using the particular customers policy. For route announcements that are generated as a result, the customer identifier is used to send it only over the customer's sessions. When one of the outgoing sessions is shared, the infrastructure provider will choose among the routes each customer is sending – most likely, a given prefix will be announce only by one of the customers (e.g., their own prefixes). The case where sessions are unique is simply a special case of the above, where a received route does not get replicated and a sent route does not have any alternatives for the infrastructure provider to choose from.

This choice to run a single routing process then impacts the data plane and general-purpose processing blades. In virtual networks, network namespaces is a mechanism to completely separate each configurable aspect of a shared packet handling process (whether in hardware or software) such that each control plane can configure their view of the data plane without any contention [13][3]. It does this through the use of contexts which essentially use the identifier of the control process as an index into the various data structures that are used in processing a packet (e.g., the forwarding table). With our proposed router platform, there is one key difference – the context is exposed to the control plane software (routing software or the software presenting configuration options to the customer's program), rather than being enforced with virtualization. The implication of this is that the control software is not tied to the strict notion that there are N completely isolated routers, and N completely isolated forwarding tables, which can lead to optimizations.

5 Simplified Management of Private Networks

A side effect of the router platform model is that it simplifies management of private networks.

Automate configuration : As discussed in Section 4, we envision that the decoupling of who owns and maintains the equipment from who is running and configuring services on the equipment will require a great deal

[2]Due to BGP not being totally ordered, multiple routes may have to be exchanged.

[3]NetNS is a Linux specific implementation, but the concept has general applicability.

of automation – involving the infrastructure provider to handle every change each customer makes is too time consuming. The network operator of a private network will benefit from this, as they too can specify using the abstract single router and leave the management of the physical infrastructure to automated tools. Additionally, the operator will be able to dynamically modify their configuration, which will have an effect on the underlying physical infrastructure.

Manage separate services independently : Additionally, the support in the infrastructure for multiple concurrent services to be run and configured independently simplifies the management of the network as a whole. Rather than having to run a new service on a test network and then gradually integrate it into the production network, each service can remain independent and co-exist on the production network without affecting one another. Further, different business units within the same company can manage their service independent of one another.

The private cloud analogy : This simplified management has an analogy in the field of cloud computing where a similar decoupling has occurred. There, software has been, and continues to be, developed to enable infrastructure to be deployed dynamically – adding and removing virtual servers. This has resulted in enterprises adopting these tools for their own internal infrastructure, commonly called a 'private cloud.' Of course, here the meaning of infrastructure provider refers to the infrastructure being maintained by one business unit (the private cloud IT staff) and the customer refers to the various business units which are independently running services on the infrastructure. ShadowNet [6] is one such infrastructure being deployed at AT&T that is being used as a testbed for testing new services. It has an ultimate goal of being used for deploying new services, which, once it reaches that goal, it will essentially be a 'private cloud' (but with the IaaS model - so it has some benefits, but not all).

Outsourced IT Many small companies cannot afford the dedicated staff needed to manage their routers, yet need routers as normal course of business. This will become increasingly true as routers support more than just connectivity (as exemplified by the many innovative applications showcased in Cisco's application development competition, many of which were geared towards enterprises). Using the router platform, the ISP or 3rd party can simply provide a configuration of a router and automated tools map that to the customer's physical infrastructure.

6 Conclusions

Like the shift that has taken place in computing towards hosted and shared architecture, we argue that the same will take place for networking, both in terms of business models and in terms of management of a private network. This will enable innovation, simplify management, and increase customer choice. We presented the router platform model where each customer is able to configure a single router (defined to include in-network general purpose processing). This single router is then automatically combined with other customers' routers and mapped onto the distributed resources available. Moving to a new model will certainly be a challenge. We hope that the benefits of doing so will make it a reality.

References

[1] D. Taylor and J. Turner, "Diversifying the Internet," in *IEEE GLOBECOM*, November 2005.

[2] N. Feamster, L. Gao, and J. Rexford, "How to lease the Internet in your spare time," in *ACM Computer Communication Review*, January 2007.

[3] G. Schaffrath, C. Werle, P. Papadimitriou, A. Feldmann, R. Bless, A. Greenhalgh, M. Kind, O. Maennel, and L. Mathy, "Network Virtualization Architecture: Proposal and Initial Prototype," in *Workshop on Virtualized Infastructure Systems and Architectures (VISA)*, August 2009.

[4] L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet Impasse Through Virtualization," in *Proceedings of the 3rd ACM Workshop on Hot Topics in Networks (HotNets-III)*, November 2004.

[5] "The Global Environment for Network Innovations (GENI)," February 2010. `http://www.geni.net`.

[6] X. Chen, Z. M. Mao, and J. V. der Merwe, "ShadowNet: A Platform for Rapid and Safe Network Evolution," in *USENIX Annual Technical Conference*, June 2009.

[7] 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.

[8] Y. Wang, M. Schapira, and J. Rexford, "Neighbor-specific BGP: more flexible routing policies while improving global stability," in *ACM SIGMETRICS*, June 2009.

[9] K. Lakshminarayanan, I. Stoica, and S. Shenker, "Routing as a Service," Tech. Rep. UCB/CSD-04-1327, UC Berkeley, January 2004.

[10] A. Edwards, A. Fischer, and A. Lain, "Diverter: A New Approach to Networking Within Virtualized Infrastructures," in *Workshop on Research on Enterprise Networking (WREN)*, August 2009.

[11] T. Wood, A. Gerber, K. Ramakrishnan, P. Shenoy, and J. V. der Merwe, "The Case for Enterprise-Ready Virtual Private Clouds," in *Workshop on Hot Topics in Cloud Computing (HotCloud)*, June 2009.

[12] E. Keller, J. Rexford, and J. V. der Merwe, "Seamless BGP Migration With Router Grafting," in *Proc. Networked Systems Design and Implementation*, April 2010. To appear.

[13] "Linux Containers - Network Namespaces." `http://lxc.sourceforge.net/network.php`, 2010.