Applying Web Principles to the Network

Brian Field, PhD
Comcast Cable Communications

Abstract

The web has been a tremendously successful application within the Internet ecosystem. There are many reasons for this success, but one key reason is likely a result of the use of open source technology. Specifically, the web is built on an open protocol (HTTP) running on an open source application (Apache) on an open source operating system (Linux) running on common, -off-the- shelf hardware. This ecosystem is ideal for innovation—any application developer can test and introduce changes into any of these software domains as appropriate for their specific application. The better changes get rolled into the ecosystem and become available for all downstream developers to use. This pay-it-forward ecosystem is likely a primary reason for the success of web applications.

The web and the Internet run over a network (routing) infrastructure that has clearly provided the foundation for making the web and Internet successful. However, the network ecosystem is not as open as the web ecosystem and this could be stifling innovation in the network space. This paper explores how we can make the network ecosystem more open and provides insights into the value this openness provides to both the network operator and the application developer.

INTRODUCTION

The web has been a tremendously successful application within the Internet ecosystem and there are many reasons for why. The roots of this success have their origins in the open source nature of its ecosystem. Specifically, the web is built on an open protocol (HTTP) running on an open source application (Apache) on an open source operating system (Linux) running on common, off-the-shelf hardware.

This ecosystem is ideal for innovation, as an application developer can test and introduce changes into any of these software domains as appropriate for their specific application. The better, more useful changes get rolled into the ecosystem and become available for all developers to use. This pay-it-forward ecosystem is likely a primary reason for the success of web applications.

Conway's Law

In late 1968, Melvin Conway proposed that "organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations" (1968).

One way to think about what Conway suggested is illustrated in the following analogy. Consider a company that consists of two divisions—one division that makes delicious peanut butter and the second division which makes delicious chocolate. If these two divisions do not have meaningful discussions or interactions, it is unlikely that they will collaborate and make a great confection known as "peanut butter cups".

Current service provider organizations are often aligned where the network engineering teams work in one portion of the company and the application engineers work in a different organization. This separation not only exists on paper, but in where these organizations are located: on different floors, buildings, even cities. Applying Conway's thinking, this logical and physical partitioning is a barrier to collaboration, cross pollination of ideas, and innovation.

This paper suggests that there is great value to be had in the network space if these separate teams were to collaborate, leveraging the technologies that have made the web so successful and applying them to the network space. Specifically, in this paper we identify and apply three web component technologies to the network space and explore the resulting value in such an application.

OPEN SOURCE AND THE WEB'S SUCCESS

As discussed earlier, one of the primary reasons for the web's success is its open ecosystem. This open environment allows the developer to make changes to accommodate specific application requirements at any of the three software layers of the web stack, shown in Figure 1. Good and great ideas often get rolled into the open source package and become available to the entire community. This is an incredibly powerful development environment as it allows innovation and agility at all layers within the ecosystem.

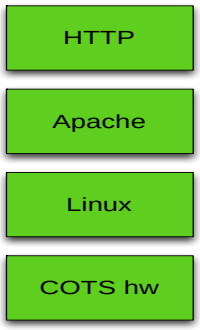

Figure 1: Web stack

Web Model Versus Router Model

Now, consider the router stack. The router stack consists of open protocols (BGP, ISIS, PIM, etc.), but these protocols are managed within the confines of a proprietary command line interface, running within a proprietary operating system, inside a proprietary chassis, running on proprietary hardware ASICs, as shown in Figure 2.

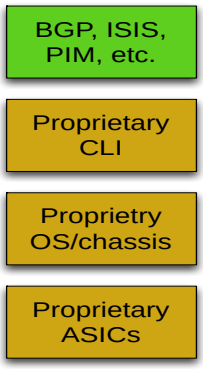

Figure 2: Today's router stack

While the IETF has provided a forum for discussing the details of routing protocols, network operators are limited in their ability to individually experiment and innovate with new concepts due to this existing proprietary router environment. In fact it has been this restrictive ecosystem that has been a motivator for OpenFlow(see Openflow reference), where a set of APIs have been developed to manage this proprietary ecosystem. These APIs are considered by some to be "SDN" (software defined networking), as shown in Figure 3.

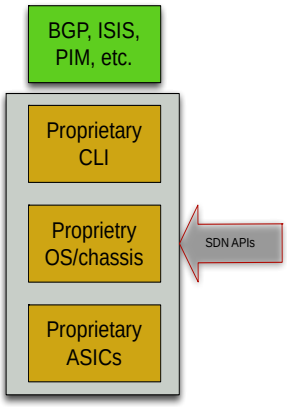

Figure 3: Is this SDN?

But is this really SDN? Or posed differently, is this all we should expect from SDN or can it leveraged for more? The service provider and MSO community may see greater benefits in the network space if we can get the router model to really align with the open source model that has made the web environment so incredibly successful.

To set the stage for this, a discussion is provided about the current service provider ecosystem and how the application technology is being deployed; this really becomes the foundation for an open environment and an innovative ecosystem in the routing domain.

Virtualization in the MSO space

Many MSOs and Service Providers are in the process of augmenting their network infrastructure with data centers full of servers that provide a virtualization environment for applications. Over the last several years it has been common for the application developers to deploy their services on virtualization platforms rather than application-specific hardware platforms. Service providers, including MSOs like Comcast, have been deploying these data center based virtualization platforms to provide an Agile environment for the deployment of their own internally created and managed applications. These virtualization platforms have gone from a centralized deployment, where there are a small number of data centers nationwide, to a more regional deployment, where data centers are located in the regions they serve. One could envision that over time these virtualization platforms will reach yet further into an MSO's footprint, including into many hub sites.

Network Topology for Applications

In simple terms, a service provider or MSO network consists of three pieces: 1) the core network, made up of core routers, with a primary purpose of forwarding packets over very large bandwidth links; 2) the network edge, consisting of routers that provide application/edge features and physical aggregation; and 3) the applications themselves.

The edge routing platform tends to have application-specific features and configurration to support constructs such as ACLs, DSCP policies, QoS settings, etc. As applications change their behavior and what they require from the network, they place new feature demands on the edge router software. As such, the code running on these edge platforms tends to need more frequent code updates. Since these edge platforms are proprietary systems built by vendors, vendors must be convinced to reengineer the software to support any new network feature that applications require. This often means getting the feature on the vendor's roadmap and waiting months or years for delivery.

APPLYING WEB PRINCIPLES TO THE ROUTER ECOSYSTEM

First Application: Router Platform Edge Virtualization

The edge router environment is one where virtualization and open source have a useful applicability. As discussed previously, edge router functionality is required to change rapidly based on application needs. Because of this, edge routers would be well served to run inside a virtual instance. Specifically, there would be great benefit in taking the existing edge router platform and partitioning it into two components: the software component that needs to evolve and change quickly, and the hardware component that performs aggregation and packet forwarding. The edge router software component could be moved into a VM in the virtualization server. Furthermore, router software could come from a router vendor or could be open source based. The open source router paradigm is particularly compelling in that it enables the service provider to innovate, be Agile, and create application-specific features in the routing code but on the service provider's timeframe. Figure 4 details the evolution of the edge router into this new VM-based router paradigm.

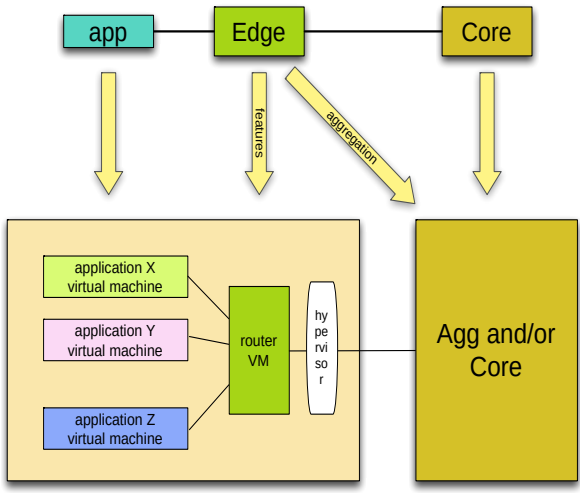

Figure 4: Virtualizing the edge router

In this model depicted in Figure 4, applications exist as VMs, but instead of having connections to the physical edge router, they have virtual Ethernet connections to this new edge router VM. This edge router VM has the set of features found in the non-virtual edge router, but these features are executed within a VM as opposed to operating within a proprietary platform and system.

Result of Virtualizing the Edge Router

Consider this new router VM paradigm, where we have moved the edge routing features into a router VM running in our virtualization platform. The resulting virtualized edge router stack looks nearly identical to the web stack depicted in Figure 5. The virtualized router stack runs as an open source VM on an open source operating system (Linux). This implementation should fundamentally improve the edge routing environment for the same reasons that the web environment benefited: this open source router VM ecosystem will provide platform innovation and agility in the router edge feature space.

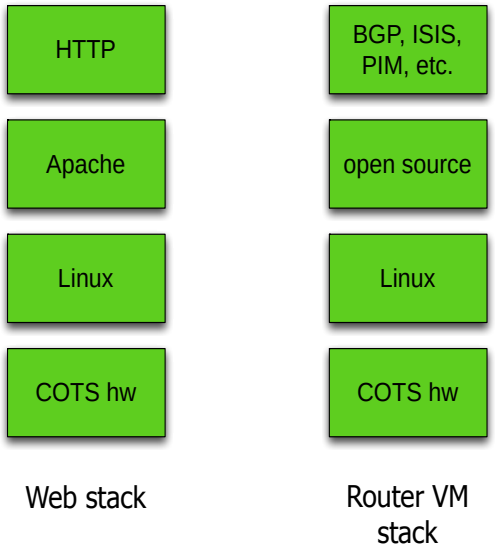

Figure 5: Web and open source router VM stack

The State of Network Control Protocols

A second area of the network ecosystem that might benefit from web ideas and concepts is in the area of network control protocols, specifically BGP. This paper focuses on BGP Link State (Gredler, 2012). BGP Link State is a new address family defined in BGP that carries details about the underlying network topology. BGP Link State encodes the state of the Interior Gateway Protocol (IGP) and makes this information available to applications. It has been proposed that BGP Link State carry not just the network topology, but other topology information, including how VMs are connected into the routing infrastructure and the power topology (what PDUs are being used to power each power supply in each server and router). The VM and power topologies are real-time exposed via LLDP, meaning that BGP Link State will likely carry both IGP information and the contents of LLDP messages (Field, 2013).

Current thinking in how to encode the IGP information into BGP is to follow the existing BGP paradigm: encode the IGP information into BGP using binary protocol constructs,

namely, a number of TLVs based on bit and byte structures.

The drivers for BGP's binary encoding system are in part to make the protocol network and processing efficient, which was a valid engineering trade-off for the network ecosystem of the late '80s and early '90s when BGP was first being developed. However, that environment of T1 links and very limited router CPU processing is long behind us. Because BGP would be required to carry very different types of information for three different topologies, more flexible encoding approaches should be considered, approaches that enable agility and innovation.

Second Application: Network Control Protocols

When considering encoding in the web environment, much control and data content is encoded as JSON. While not as compact as BGP binary encoding, JSON encoding is well known and commonly used in the application space, while writing binary-based BGP encoding is relegated to handfuls of network engineers with BGP expertise. To take advantage of the benefits that come from wide adoption and expertise, JSON encoding should be seriously explored for new address families for BGP, such as Link State.

In addition, if encoding the BGP Link State content as JSON has benefits, then consider the possible benefits of refining the BGP data passing primitives (OPEN, UPDATE, NOTIFICATION, etc.) to leverage those that are commonly used in web-based technologies. Specifically, make the BGP protocol RESTful.

In summary the second application of web principes to the network is that new BGP address families use JSON encoding and RESTful primitves. This basic step could be a key enabler to innovation and agility in the network control plane space and be key to empowering applications to make the most of the network.

What Is the Right Network Topology?

In the past, much of the job of a network engineer was to understand where an application would physically plug into the network, how much bandwidth it might consume, and with what end-points this application would communicate. The network engineer would then consider different failure scenarios and, based on this, determine how much capacity was needed in each portion of the network. This task was relatively deterministic in the sense that a network engineer would know with a high level of certainty where an application would exist in the network.

This paradigm does not necessarily apply in an environment where applications can easily be spun up in any corner of the virtualized cloud network. The days of engineering a "custom" network design are behind us. Instead, in the network space engineers and developers need to move to an "instrumentation over engineering" mentality. The general premise in this approach is that no matter how much engineering analysis goes into understanding application traffic patterns as it relates to the physical netwotk topology, the network topology will be inefficient as applications change their requirements, operation, and move around the cloud infrastructure. Therefore, a prudent path to take is to enable much more detailed instrumentation of the network—specifically, provide a much more granular and measured understanding of what each virtualized application is talking to. In a virtualized application environment, one might suggest that a network topology is not inefficient when it runs out of capacity in a portion of the network, but rather that the placement of the application VMs might be less than optimal on the existing network topology.

The shift in thinking proposed here is very much is inline with what the web development community has exceled at—namely analytics.

Analytics

In the web space, analytics are a key part of understanding how well an application is operating and understanding who is using the application. This is done by performing a detailed analysis on the application's logs. In general, each transaction is logged and processed, and relevant information extracted. From this information, the application developers and operation teams can refine the application accordingly.

In the network space, these analytics are performed by mechanisms such as NetFlow. NetFlow provides detailed information about high-level traffic patterns. However, nearly all NetFlow information is sampled, meaning only a very small subset of all packets are actually recorded and processed. Given the performance concerns with attempting to gather too much NetFlow data (and hence overwhelming the routers), sampling one packet per several thousand forwarded is common. In the web space, it would be nearly unheard of to log only one transaction out of several thousand performed.

Given this, per packet analysis is the third application of web principles to the network proposed here.

Third Application: Network Analytics

Recall that the first application of web principles to the network was to drive an open source router instance that can run in a VM. This VM then becomes the interconnect point for virtualization applications. This paper proposes that one of the first features embedded into this router VM software is per packet network analytics.

Specifically, the per packet analytics should track the distance each packet is traversing in the network. This might be grouped into distances such as:

- Traffic that is local to this server
- Traffic that is local to this site
- Traffic that is local to the metro area
- Traffic that is local to the service provider
- Internet traffic

This proposed grouping is depicted in Figure 6.

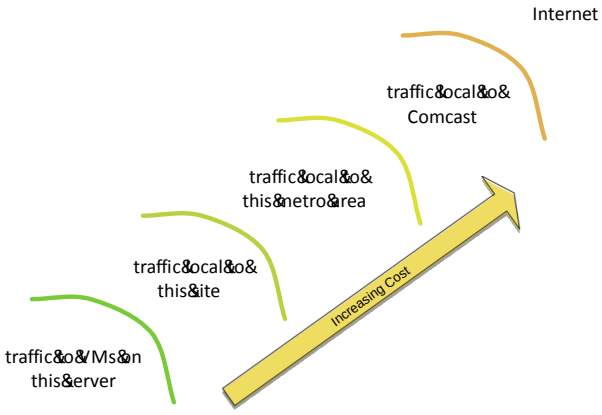

Figure 6: Per packet network analytics

At a minimum, the number of packets and bytes being sent to and from each of these logical domains over narrow time horizons would be tracked. In addition the address family (IPv4 or IPv6), the size of the packets, and possibly other aspects that will help service providers understand the bandwidth-distance usage of this application could be tracked. Why is knowing this information useful? Because rather than attempting to re-architect the network when there is insufficient network capacity—an activity that likely takes weeks and months to complete—service providers may be able to better leverage the existing network resources by re-shuffling the location of the application VMs. By having this per packet level bandwidth-distance usage information over narrow time windows for each application VM, the service provider can derive a very detailed understanding of the impact that moving an

application VM will have on the existing network location and to the VM's new network location.

PROGRAM THE NETWORK OR THE PACKET?

The focus on this paper has been applying web principles to the network and correspondingly making the network eco-system more open, agile. and a platform for innovation. Much of the work in the SDN space has been related to "programming the network"; namely, installing network state data from entities outside of the classic router control plane. This clearly enables innovation between the application and network domain, where the application is able to install specific state data into the network. However, when considering router and forwarding technology and evolution, putting additional state data in the router forwarding plane consumes ASIC memory and depending on the amount of state data injected, could place limiting factors on cost effective ways for the router ecosystem to evolve.

An alternative mechanism, called segment routing has been proposed that provides a more flexible way to steer traffic through the network and application space (see segment routing reference). Rather than put state into the network to define a forwarding path for an application, the state instead is inserted into the packet's header (via an IPv6 extension header) by the application or network router for the desired set of application packets. In the segment routing model changing the path that an application's packets takes through the network is done simply by changing the information in the packet extension header, rather than reprogramming state across a number of router devices. Having an open source router edge has the potential to accelerate the deployment of segment routing technology within a service provider or MSO network.

VIRTUALIZATION IS THE ENABLER

The technology that is enabling the approach outlined in this paper is virtualization. This approach takes the virtualization paradigm developed and used extensively by applications and leverage it in the networking space.

Specifically, a proposal to take the edge routing functionality out of the legacy edge router and put it into a VM has been discussed. As part of this migration, the open source ecosystem will be driven to develop production-ready instances of routing code. This open source paradigm then enables innovation and agility in the routing space much like is done in the application space today. Service providers are no longer tied to the development cycles of the existing router vendors and a service provider can innovate on their own technology and timeframes.

The second evolutionary step proposed moving network protocols from their binary format to a format that is more readily extended and where existing tools and software paradigms exist to easily process and develop against. Specifically, we suggest that network protocols, such as BGP Link State, encode information in JSON format and use RESTful primitives.

The third step we propose is to leverage the analytics and Big Data paradigms successfully used in the web space. We specifically propose moving from the "sampled" Netflow paradigm to one where we track per packet bandwidth-distance information. This information then becomes the data that is used as to determine how "inefficient" the application VM's current placement is, and then using this information, VMs can be reshuffled to make better use of the physical network resources.

Is this SDN?

Is the proposed evolution of the network space, as driven by applying web concepts to the network that outlined here SDN? Yes, it is SDN—or minimally it addresses a portion of the SDN problem space. Specifically, the thinking outlined in this paper enables both innovation and agility in the network space by both network operators and application developers, and these are key features in many SDN paradigms.

SUMMARY

In this paper, a number of characteristics that have made the web a success are considered, including an open source paradigm that enables innovation, recasting network protocols to operate using the encoding and primitive mechanisms that are widely used for web applications, and to embrace a paradigm of instrumentation over detailed network engineering via network analytics.

References

Conway, Melvin E., "How Do Committees Invent", *Datamation Magazine*, April, 1968.

Field, Brian, "Exposing network, VM edge and power topology via LLDP and BGP-LinkState – and possible implications", NANOG, June 2013, http://www.nanog.org/sites/default/files/tues.general.field_.topology.32.pdf

Gredler, H., et al, "North-Bound Distribution of Link-State and TE Information using BGP, July 15, 2002, http://tools.ietf.org/html/draft-gredler-idr-ls-distribution-02

Openflow reference: https://www.opennetworking.org/

Segment-routing-reference

https://datatracker.ietf.org/wg/spring/