Network Programmability is the answer!

What was the question again?

Or, is there really a case for Network Programmability?

Jacobus Van der Merwe and Charles Kalmanek
AT&T Labs - Research

Recent developments in the broader networking community involve the decomposing and/or refactoring of network functions in order to ease the introduction of new protocols and services and indeed new networking architectures [3, 4, 8, 6, 5]. At first blush this appears to be a most desirable development from a service provider point of view, because it enables the introduction of new features/services without the need for industry wide standards or even buy-in from vendors. However, to some extent, as a community we have been here before: recall the network programmability efforts of the late 1990's [11, 9, 12]. These earlier efforts met with limited (if any) success. While many reasons can be found for this less than desirable outcome, we argue that a primary reason (if not *the* primary reason) for the lack of success resulted from a poor understanding of the problems that these approaches were meant to solve. Indeed, as technologists, we exhibit a proneness to focus on the often more clearly defined mechanistic side of a given problem, rather than questioning whether that is the *right* problem to address. Despite constant customer demand for new services/features, and the obvious monetary incentive associated with that, the introduction of new network services and features has been an extremely protracted process for ISPs. We argue that the *difficulty to rapidly and safely deploy new services and service features* is the single most important challenge facing ISPs today. Given this position, we first define what we mean with network programmability and network services and then attempt to articulate why service deployment is challenging in today's environment. Finally, we offer some thoughts about the applicability (or not) of open/programmable architectures to address these problems.

What do we mean with open/programmable? We define an open or programmable architecture as a network device (e.g., a router platform) that allows third party code to be executed on the device. We further refine this definition to explicitly include the means to provide such third party functions in both the control plane and the data plane.

What do we mean with "network service/feature"? Broadly speaking we can define a new network service and/or network feature, as *any* change in network functionality, or in the way the network is operated. This broad definition would indeed capture most (or all) services in a provider network, i.e., from upgrading the transport network from OCxx to OCyy, to deploying network monitoring systems or overlay nodes to enable content distribution. For the purpose of this write up we narrow the scope of applicable network services/features as those that operate at the network layer or above. While by no means exhaustive, below we provide a few example services/features categories for illustrative purposes:

- *"Network wide" protocol stacks:* The most challenging new services are those that involve the deployment of complete new protocol stacks on all (or most) network elements. Examples include the deployment of MPLS, MPLS-VPNs and IPv6 all of which are realizations of well defined standards that have been developed by the vendor and provider communities, typically over extended periods of time.

- *New ways to manipulate existing protocols:* On the other end of the spectrum in terms of potential "disruption impact", are services or service features that allow existing protocols to be manipulated in a new way. An example would be the use of community values to allow customers to signal the desired preference value of a route to the provider (RFC 1998). Another example would be the RCP/IRSCP route control work [3, 13, 14] where a separate control element manipulate route selection in routers, without requiring any changes to the routers.

- *New "localized" protocols:* The networking research literature is rife with new protocols or extensions of existing protocols, which might require fairly localized functionality changes to enable new services/features[1]. One example is the MIRO multipath routing protocol which allows ASes on a bilateral basis to reveal and use routes other than the normal single path selected and used by BGP [15].

- *"Cross-layer-aware" services:* While the layering abstraction of communication protocols has been an undeniable cornerstone of network development, new network applications put increasing pressure on network services to be "cross-layer-aware". For example, most first generation content distribution networks (CDNs) take network latency into account, but other than that is mostly unaware of the underlying network [2]. This approach works well when the volume of content is small relative to the underlying network and other traffic in the network. However, when the content volume constitutes a significant fraction of the underlying network capacity, it becomes important to be aware of those constraints at the content distribution level. Ensuring the continued operation of realtime and business critical applications in the face of planned and unplanned outages, presents another domain where cross-layer visibility becomes critically important [10].

Why is service deployment hard? Here we attempt to provide some rational for why the deployment of new services present such a challenge for providers by articulating some of the inherent difficulties with providing new network services:

- *Vendor support:* In the absence of an open/programmable environment lack of vendor support for a particular feature and/or protocol

[1] These approaches are mainly constrained to an "existence" in the research literature and would clearly benefit most from an open/programmable architecture.

is clearly prohibitive for any service deployment relying on such a feature or protocol.

- *Impact on existing services:* Because networks are inherently shared, any new service or service feature that gets deployed can potentially have an impact on existing services. One (historical) example would be the multi protocol extensions to BGP to enable MPLS-VPNs. Bugs in new BGP software that is capable of this new protocol extension clearly has the potential to impact existing IPv4 BGP interactions. Another example would be inherent dependencies between protocols, such as the label distribution protocol (LDP) in MPLS networks, which make use of an interior gateway protocol (IGP) to determine its link state paths (LSPs). In this case problems with the IGP protocol will naturally also cause problems in the LDP protocol relying on it.

Note also that network services are normally "cumulative" in the sense that once deployed and used, network services are very rarely "switched off". This means that over time the dependencies and the potential for negative impact increase rather than diminish. It is this inherent fate sharing that differentiate network based services from services deployed in end-systems connected to networks. In the latter case, because of inherent separation of different end-systems, services can be built in relative isolation without fear of impacting (or being impacted by) other services. The net effect of the inherent fate sharing of network services is that new services have to undergo extensive testing to reduce the risk of negatively impacting existing services.

- *Support systems:* All network services require some auxiliary support systems including: (i) configuration management systems (new services need to be configured typically across many network elements), (ii) network management systems (network elements and protocols need to be monitored and maintained), (iii) service monitoring systems (for example to ensure that network wide service level agreements, e.g., loss and delay, or video quality, are being met), (iv) provisioning systems (e.g., to ensure the timely build out of popular services), (v) customer databases and portals (for customer facing services, customer information needs to be maintained, while customers need to be provided with "customer views" of the service), (vi) billing systems (to stay in business customers need to be billed for the services they use). While not all network services require all of these support systems, or unique versions of the support systems, all new network services require some new support system or new features to an existing support system.

The role of open/programmable architectures. We now consider the three challenges to network service deployment listed above in turn and examine the potential role of network programmability to curtail each challenge.

Overcoming lack of *vendor support* is clearly the main benefit to be had from an open/programmable architecture. In a truly open environment, lack of a protocol or protocol feature can be addressed by service providers themselves, or by third parties that provide specialized solutions (that might not be of interest to the vendor). There are potential issues with such an approach though, for example, when provider or third party code is allowed to run on vendor platforms, the danger exist for finger pointing because of the lack of clearly defined boundaries of accountability.

In terms of *impact on existing services* it is not clear that an open/programmable architecture by itself will make the problem any easier. Indeed, the potential for third party code to interfere with existing protocol stacks seems more likely than in the case where all code is developed by the same organization. Further, making nodes programmable does not remove the dependencies/fate sharing inherent with different network services in the same network. We note that the partitioning afforded by virtualization appears to be the most promising approach to address these issues. Specifically, end-system based virtualization techniques [1] can be utilized in the control plane, while modern routing platforms already provide very sophisticated virtualization mechanisms in the data path [7]. Further, the "carrier supporting carrier" concept enabled by MPLS VPNs provide an example of how virtualization can be extended network wide.

Finally, in terms of *support systems* it is again not clear whether open/programmable architectures will per se improve the situation. However, we would argue that simplifying the development of support systems *should* be an explicit goal of any open/programmable architecture to reach its full potential. We are not advocating that an open/programmable architecture by itself will solve the support system problem, indeed we expect support systems to evolve, however, we do expect open/programmable architectures to significantly contribute to this solution.

Final word. In conclusion we would argue that there is a case to be made for network programmability to support rapid and safe deployment of network services. However, we would urge practitioners in this area to take a holistic view of what deploying network services entail and specifically considering the role of network programmability to simplify the "support systems" aspect of this picture.

References

[1] BARHAM, P., DRAGOVIC, B., FRASER, K., HAND, S., HARRIS, T., HO, A., NEUGEBAR, R., PRATT, I., AND WARFIELD, A. Xen and the art of virtualization. Proceedings of the ACM SOSP, October 2003.

[2] BILIRIS, A., CRANOR, C., DOUGLAS, F., RABINOVICH, M., SIBAL, S., SPATSCHECK, O., AND STURM, W. CDN Brokering. Proceedings of WCW'01, June 2001.

[3] FEAMSTER, N., BALAKRISHNAN, H., REXFORD, J., SHAIKH, A., AND VAN DER MERWE, J. The Case for Separating Routing from Routers. FDNA Workshop, Aug 2004.

[4] FEAMSTER, N., GAO, L., AND REXFORD, J. How to Lease the Internet in Your Spare Time. Compter Communications Review - Short technical note, January 2007.

[5] GENI. Global Environment for Network Innovations. `www.geni.net`.

[6] GREENBERG, A., HJALMTYSSON, G., MALTZ, D. A., MYERS, A., REXFORD, J., XIE, G., YAN, H., ZHAN, J., AND ZHANG, H. A clean slate 4D approach to network control and management. *SIGCOMM Comput. Commun. Rev. 35*, 5 (2005).

[7] JUNIPER NETWORKS. Configuring virtual routers. JUNOSe 8.0.x Systems Basics Configuration Guide, 2006.

[8] LAKSHMAN, T., NANDAGOPAL, T., RAMJEE, R., SABNANI, K., AND WOO, T. The SoftRouter Architecture. ACM HotNets-III Workshop, November 2004.

[9] LAZAR, A. Programming telecommunication networks. *IEEE Network* (September/October 1997).

[10] RAMAKRISHNAN, K., SHENOY, P., AND VAN DER MERWE, J. Live data center migration across wans: A robust cooperative context aware approach. Submitted for publication, May 2007.

[11] ROONEY, S., VAN DER MERWE, J. E., CROSBY, S., AND LESLIE, I. The tempest, a framework for safe, resource assured, programmable networks. *IEEE Communications 36*, 10 (October 1998).

[12] TENNENHOUSE, D., SMITH, J., SINCOSKIE, D., WETHERALL, D., AND MINDEN., G. J. A survey of active network research. *IEEE Communications Magazine 35*, 1 (January 1997).

[13] VAN DER MERWE, J., AND ET.AL. Dynamic Connectivity Management with an Intelligent Route Service Control Point. ACM SIGCOMM Workshop on Internet Network Management (INM), October 2006.

[14] VERKAIK, P., PEI, D., SCHOLL, T., SHAIKH, A., SNOEREN, A., AND VAN DER MERWE, J. Wresting control from bgp: Scalable fine-grained route control. To be published: USENIXAnnual Technical Conference, June 2007.

[15] XU, W., AND REXFORD, J. MIRO: Multi-path Interdomain ROuting. Proceedings of ACM Sigcomm, October 2006.