Collaborating with the Enemy on Network Management

Chris Hall[2], Dongting Yu[1], Zhi-li Zhang[4], Jonathan Stout[3], Andrew Odlyzko[4], Andrew W. Moore[1], Jean Camp[3], Kevin Benton[3], and Ross Anderson[1(✉)]

[1] University of Cambridge, Cambridge, UK
Ross.Anderson@cl.cam.ac.uk
[2] Highwayman Associates, Leatherhead, UK
[3] University of Indiana, Bloomington, IN 47405-7000, USA
[4] University of Minnesota, Minneapolis, MN 55455, USA

Abstract. Software Defined Networking (SDN) deconstructs the current routing infrastructure into a small number of controllers, which are general purpose computers, and a large number of switches which are programmable forwarding engines. It is already deployed in data centres, where it offers considerable advantages of both cost and flexibility over a switching fabric of traditional routers. Such applications have a single controlling organisation and issues of trust between subdomains do not really arise. However for SDN to fulfil its potential, it is necessary to design and develop mechanisms for smart networks with mutually mistrustful principals.

In an earlier paper, we used as an example an airport where we might have 100,000 staff working for 3,000 different firms which include not just competitors but also organisations in a state of conflict (for example, El Al and Iran Air). That paper discussed using hierarchical control structures to delegate trust with mechanisms focussed on preventing denial-of-service attacks, with the assumption that confidentiality and integrity would be provided by the principals at higher layers. But this turns out to be a quagmire. Can you run your app and your enemy's app on the same controllers of the same fabric, and get a passable separation of behaviour on private networks that run over the same switches? And can all this be done without a trusted root anywhere?

This paper reports a project to build a test environment that adapts Quagga so that a software defined network can be automatically configured using information learned from BGP. Our Quagga for SDN Module, "QuaSM", is designed to support the use of SDN in three further use cases: in a network exchange point, in an organisation seeking to join up two or more SDN islands using an existing BGP fabric; and in security research on virtual networking.

1 Introduction

At this workshop in 2013, we discussed the security of software defined networking (SDN) [1]. SDN is a new approach to network management, which grew out

B. Christianson et al. (Eds.): Security Protocols 2014, LNCS 8809, pp. 154–162, 2014.
DOI: 10.1007/978-3-319-12400-1_15

of earlier research into active networks [2]. It separates the control plane and data plane so that forwarding is done by simple but fast hardware, while the control logic is offloaded to commodity PCs [3]. These PCs calculate and install logic on forwarding devices, which inspect the headers of incoming packets and follow forwarding rules specifying where and how each packet is to be forwarded. The behaviour of the network (i.e. what the network does with all the packets it receives) is defined by the sum of all forwarding rules in all forwarding devices. Instead of using vendor-supplied code, operators can now use any software of their choice as the control plane, with sufficient freedom in deciding how the overall network behaves and how each forwarding device reacts to particular packets. SDN is gaining traction in industry because of simpler network management and reduced capital expenditure.

In conventional routed networks, each router has a control plane and a data plane, which in large routers are manifestly separated. However, in a router the local control software constructs forwarding rules based on its local view of the state of the network and its local configuration. Routers are configured in a variety of ways and have a variety of features to control the construction of forwarding rules — depending on the make and model of the router and, in larger routers, the hardware configuration. The control plane in a routed network is fragmented. The behaviour of the network depends on all routers having a sufficiently good view of the state of the network and each one being configured to construct the required forwarding rules. Routers talk a variety of routeing protocols to learn and distribute network state, but the information available to each router is limited by the routeing protocol. Given the limited capabilities of routers that manufacturers provide and the complexity of the task, the effectiveness of the configuration is limited.

In contrast, an SDN control plane may be integrated and largely independent of the underlying forwarding devices [4]. The revolution here is that the management of the network is "top down"; the network's policies may be fed down into the control plane, which will construct the forwarding rules — depending on the state and topology of the network — which in turn are fed down into the forwarding plane. This is the fundamental difference with a conventional routed network, which is managed "bottom up"; there the routes are managed individually, with the intention that the emergent behaviour of those independent actors will meet the network's requirements. Furthermore, the capabilities of the (open) control plane are a "simple matter of programming", unconstrained by current routeing protocols and liberated from today's tightly bound control/forwarding (closed) devices. This should lead to rapid advances in control software, as was envisaged by McKeown et al [5].

The separation of forwarding also opens the way for innovation and competition in the hardware-intensive world of the forwarding device. The immediate effect is a reduction in capital costs, which already attracts interest. Further, "middle-boxes" such as firewalls, load-balancers and intrusion-detection systems are significant parts of most networks. These devices comprise both specialised control software and specialised forwarding hardware. Assuming that forwarding devices become more general and more powerful, middle-box functions may

well be absorbed into and distributed across the SDN, which could simplify and reduce costs. Further, where firewalls in an existing network are choke points, an SDN might in theory mobilise the entire network to repel unwanted packets.

Thrilling though innovation and cost saving in the data plane may be, the real revolution that SDN drives is the revolution in network management. Most SDN deployments today are in the data centre, where many tens of thousands of servers are connected by thousands of switches, in a largely switched (rather than routed) network [6]. Here the key advantages of an SDN are that the management of the network can be more effectively automated, improving the efficiency of the network, increasing the speed and ease of implementing changes, and reducing operational costs. Switched networks are straightforward, the switches are told what to do and they do that — the problem is that changing the network means telling the switches to do something different. Switched networks will guarantee that the configured capacity is available between two points while everything is working; the problem is that they do not automatically respond to failures. Yet modern data centres use large numbers of commodity PCs rather than small numbers of expensive mainframes, and the key enabling technology is automated failure recovery. Firms like Google pioneered computing platforms that recover more or less seamlessly from the loss of a single hard drive, or a whole PC, or a rack, or even an entire data centre; automating the control plane for the vast switched networks in such centres is both necessary and revolutionary.

The next step is to take SDN beyond the closed environment of a data centre. At a carrier, for example, it can also improve the efficiency, increase the speed and ease of implementing changes, and reduce operational costs. Centralising the control plane of a large network allows it to be managed as a whole (top down). To support this, the control plane must keep track of the state of the whole network, so that the automated-decision making processes can, guided by the configuration, program the data plane to fulfil the operator's requirements to the greatest extent possible, given the state of the network at the time. This vision of complete control of the network is seductive and compelling. Clearly, however, no large real-world network is going to be controlled by a single, all-seeing, all-knowing controller device. In a data centre the SDN control plane may be implemented that way (with suitable provision for fail-over), but beyond that we must expect it to be a distributed system, perhaps based on a distributed database of configuration and network state. The control plane may be implemented as a hierarchy of systems (as we described in our 2013 paper [1]), with local systems making decisions for the local network, and regional or global systems making higher level, larger-grain decisions. The argument for SDN is that once liberated from today's routers, whose function is based on the network and computing resources of the 1970's, the control plane can be transformed.

Currently, routers keep track of the state of a network using internal routeing protocols (notably OSPF, IS-IS and iBGP). These protocols provide each router with a view of the network which is more or less complete (at least locally) and more or less timely. In an SDN we may expect this information to be maintained at the logical centre of the network, so these protocols are likely to be casualties of the SDN revolution.

Externally, where different operators' networks are connected, the routers speak eBGP, the de facto routing protocol of the Internet, to each other. There is clearly no likelihood of eBGP being sent to *Madame la Guillotine* in the near future, so if only as a transitional measure, we need to consider how to use BGP as a component of an SDN control plane.

2 QuaSM

Our Quagga for SDN Module, or 'QuaSM', is being produced as part of a DARPA seedling project. It is designed to provide a component for use in an SDN control plane.

A BGP router accepts routes advertised to it by its "neighbours". Each route comprises a "network prefix" and a set of "attributes". Each route represents an undertaking from the advertiser of the route that it will forward packets destined for the IP addresses given by the prefix towards their destination. The receiver of a route will examine it and decide whether to include it in its "Local Routeing Information Base" (Loc-RIB) as a candidate for selection, and in the process may modify some attributes. The router will select the route which it considers to be the best of the available candidates. That route will be installed for use in the router's forwarding hardware, and will be advertised to the router's neighbours. (This is not a complete description of BGP, but is sufficient for our discussion.)

Underlying the way that BGP works is the assumption that the BGP process is making routeing decisions on behalf of the device it is running on. Where relevant, the selection process will consider the device's place in the network, so the "network cost" of using a given route from the perspective of the router will be taken into account. The notion of network cost of a route is defined by its operator, and usually includes factors such as the number of hops and preference derived from business relationships.

For an SDN control plane BGP needs to be lifted out of individual routers, and become part of a logically (if not actually) centralised system. So a key part of QuaSM is to separate out the network cost considerations and to allow for multiple, parallel selection processes. A network of any size will generally comprise a number of Points of Presence (PoPs) connected together. A PoP may be a large data centre or may be a router or two connecting to customers or other networks. QuaSM views the underlying network as comprising a number of "route-contexts", and runs a separate selection process for each one. Where network cost is relevant when choosing between routes for use by devices in route-context 'C', then if route '1' is via a neighbour in route-context 'A' and route '2' is via a neighbour in route-context 'B', then the network costs of getting from 'C' to 'A' and from 'C' to 'B' are taken into account. In a given network each PoP might be treated as a route-context. A cluster of small PoPs might be treated as a route-context; a really large PoP might be treated as more than one. The essence is that a collection of devices may be lumped together in a route-context when it makes sense for QuaSM to make the same decision for all

of them, which means that the network cost between these devices is, effectively, zero. Another perspective is that routeing decisions optimise network cost over route contexts rather than over individual nodes or routes.

QuaSM manages BGP conversations with other BGP speakers, accepts routes from neighbours, maintains the usual Route Information Bases, makes the usual BGP routeing decisions (but one for each route-context), and announces routes to neighbours. The routes announced to neighbour 'C' will be those selected for the route context in which the connection to 'C' is made. The interface between QuaSM and the rest of the SDN control system includes a means to configure the BGP processing, a means for BGP to discover and be told about network costs (using an "infinite" cost to signal loss of connection), and a means for BGP to pass the selected route (one for each route context) to the rest of the SDN.

All of this is recognisably BGP. In this form QuaSM can be dropped into an SDN control plane as a straightforward replacement for an existing BGP mesh, where it manages all the eBGP connections and replaces all the iBGP ones. The interface between QuaSM and the rest of the SDN can be extended, first so that more of the information learned by BGP is available to the control plane, and second to augment or replace the selection process. Further work may well be needed to scale this approach up for large networks. As for SDN in general, for large networks it may be necessary to subdivide the network and perhaps create a hierarchy.

Another possible application for QuaSM is the use of BGP to tie islands of SDN together. In this case the QuaSM instance in one SDN island would be talking iBGP to instances in other SDN islands, and perhaps to the rest of the network's iBGP mesh.

The flexibility of SDN and operator-run code also allows for a smooth transition in the event we find ourselves connecting a traditional BGP network to an SDN-for-BGP network. The SDN part, being newer than traditional BGP, can understand and talk in the protocol language in a backwards compatible manner. Since no flag day will happen in which everyone on the Internet updates their routing software simultaneously, the ability to manage a transition stage well is crucial to the deployment of a new technology.

3 SDN and Security

An SDN control plane will comprise control and forwarding elements connected by some control network. It is clearly essential to ensure that those elements and that network cannot be suborned or prevented from working. Where the control network is separate and self-contained (as, perhaps, in a data centre), that may suffice. The use of TLS and some means of distributing the necessary certificates may suffice in the more general case, though the network might wish to give priority to control traffic where it shares bandwidth with other traffic. Those are issues which affect the security of the inner workings of the SDN, which are essential to ensure that the network is not disrupted by an enemy, but are not related to collaborating with the enemy.

Across the data plane a network will carry all sorts of traffic, but not all traffic will be welcome in all parts of the network, and some traffic may not be welcome at all. Some traffic may be welcome in limited amounts. As noted above, current firewalls, load-balancers, intrusion detection systems and so on are "point solutions". For them to be effective, network engineers must ensure that any traffic which may be suspect will pass through the required middle-boxes, and that all middle-boxes are configured correctly. This is inevitably in tension with the need to avoid or mitigate service-denial attacks[1]. The ability to specify traffic rules centrally, and leave the system to decide where and how to implement those rules, may transform a network's ability to control the traffic it carries. This could be done with existing middle-boxes: the SDN control plane could take responsibility for directing packets through the required middle-boxes. Given a way to program those middle-boxes, the control plane software could automate their configuration as well; it's worth noting that if the control plane is compromised then the middle-boxes can be avoided. When sufficiently capable and powerful general-purpose forwarding devices become available, the programming of traffic control rules may be simplified. In short, the greater control and automation implicit in SDN can transform a network's ability to manage collaboration with the enemy in handling packets. However the control plane itself is of necessity trusted, so it must run on a trustworthy platform.

Network Ingress Filtering (BCP38/RFC2827 and 2267 before it) has been recommended practice for some 15 years, but is not commonly implemented. If all networks policed incoming packets, and rejected those with spoofed source addresses, then some forms of Denial-of-Service attack would be impossible and others at least traceable to their source. Unfortunately, policing incoming packets is not straightforward and there is the usual lack of incentive to overcome that. Greater automation does not solve the incentive problem, but could make the implementation straightforward, and perhaps reduce costs to the point that it becomes the default option.

Security concerns with BGP are not new. A more secure form of BGP has been at least fifteen years in the making. The latest and one that is most likely to see real world deployment, is BGPSEC, which has spawned some 22 RFCs in the last two years, though the "BGPSEC Protocol Specification" is still in draft form. BGPSEC sets out to secure the BGP AS_PATH attribute, so as to make it possible to detect most forms of path tampering. However, BGPSEC does not address "route leaks", which are the most common way of disrupting global routeing; nor does it mitigate those attacks that do not require a forged AS_PATH. The processor and memory requirements projected for BGPSEC are well beyond the capability of current routers, even high-end million dollar devices. Furthermore,

[1] Firewall rules are not straightforward, and it may be necessary to configure each firewall differently: where not all the firewalls in the network are of the same make and model; or where for performance or other reasons not all rules for the network can be installed in every router; or where the rules for different parts of the network simply aren't the same; and so on.

the security economics of deployment are difficult, as the protocol's benefits are more global than local.

Lifting BGP out of routers may yet prove to be the most practical way to implement BGPSEC. Bringing the BGP routeing information and decision making into the light will also allow for innovation in the verification of the information and allow other sources of information to influence route selection. A useful analogy might be drawn here with authentication, where for years researchers at this workshop (and elsewhere) have focussed on password verification protocols. Nowadays, user authentication is becoming a 'Big Data' service; firms such as Google, Facebook and Microsoft see billions of authentications a day and are far better placed than individual e-commerce websites to spot an account compromise. Large-scale analytic techniques can also be used for network security tasks, once a suitable platform can be deployed. Further improvements to the security of inter-domain routeing need not then take another fifteen years!

Integrating the information learned by BGP with the rest of the SDN network state can allow the control plane to create low level forwarding rules consistent with the high level routeing. For example, packets sent to a given neighbour should be destined for addresses which the neighbour has announced valid routes for, and those addresses only. Conversely, packets received from a given neighbour should only have source addresses for which the neighbour has announced valid routes.

In addition, most network failures at present result from operator error, as traditional routers are managed using 1970s-vintage command-line interfaces; these do not support such desirable features as atomic updates and managed rollbacks. Worse, the router vendors have all customised the commands slightly making it easy for even experienced operators to make mistakes. Modern user interfaces with proper tools can improve reliability and usability as well as security.

4 Latency Rains on Parade?

The "logically centralised" SDN control plane appears to promise the nirvana of complete, automatic control of the network. Instead of having to guess how a network of independent routers will respond to changes in traffic or link or equipment failure, the SDN control plane, armed with perfect and complete knowledge will re-optimise the network.

The most obvious issue with this is latency. In a global network, a device in Sydney is 150 ms from London, as the packet flies. So a network event in Sydney would take at least 150 ms to be registered in London, and any network changes would take at least 150 ms to make their way back. This does not make obvious sense. In fact, when it comes to latency, engineering reality is often very much worse than the limits set by physics, and the main reason is the needless introduction by engineers at many levels in systems and networks of mechanisms that introduce unnecessary delay [7]. Locating controllers at such a distance from the switches they drive may degrade telepresence and other services that require interactivity [8].

Scaling and latency point towards the need for an SDN to be subdivided. Latency points to a subdivision on a geographical basis, so that local decisions can be made locally. Unsurprisingly, perhaps, this returns the SDN to some of the problems with existing routed networks, where latency and information skew cause, transient, routeing issues. How very large networks will solve these issues remains to be seen. So far there have been some proprietary solutions; Google's implementation is described at [6].

5 The Research Opportunities

Software defined networks are the new cutting edge of networking innovation. They are widely deployed in large data centres, where they first save cost by replacing expensive routers with commodity hardware, and second provide resilience by supporting intelligent failover to replace failed machines or racks. Deployment is starting in Internet exchange points (we have worked with one IXP in developing the BGP-for-SDN module software) [9]. The next likely target after that is in corporate networks, which already have islands of SDN that they wish to link up; and in large carriers, where the driver will be saving labour costs. More complex multitenanted networks, such as the airport example discussed in [1] here last year, may follow.

SDN technologies have the potential to deliver much more secure networks, by making practical the deployment of security protocol suites such as BGP SEC; by enabling network-wide monitoring, analytics and control in order to deal with the threats that BGP SEC ignores; and by allowing specific services such as intrusion detection, DDoS prevention, firewalls and indeed interception to be re-engineered as network applications. This creates a lot of scope for creative innovation in defence, and (it must also be said) in attack. As much of the design and development work for large-scale SDN remains to be done, there is an opportunity for security researchers to get involved in time to make a difference.

We have therefore been developing (in the context of a DARPA seeding project) a Quagga SDN Module, QuaSM, that enables researchers and developers to carve out the BGP functions from an SDN network for the purposes of experimentation and testing [10]. This will enable SDN to build on the existing BGP mechanisms for negotiating the details of transit and peering between mutually distrustful parties. By using BGP as a scaffolding, we can not only build the next generation of production systems for slightly more complex and decentralised environments, but provide a platform on which researchers can experiment with novel trust mechanisms for virtual networks. Once we have two separated networks that interoperate using BGP, we can also test two logically separate networks that interoperate using BGP while running on the same switch fabric. It will then be possible for security researchers to play network games by combining ideas from DDoS, red pill/blue pill, concurrency/API and all sorts of other attacks, to determine empirically whether a network can obey the maxim 'hold your enemies close'.

Acknowledgement. The work described in this paper was funded under DARPA BA 12-29 FA8750-13-2-0023, 'Hardening the next generation control plane', whose support is gratefully acknowledged.

References

1. Yu, D., Moore, A.W., Hall, C., Anderson, R.: Authentication for resilience: the case of SDN. In: Christianson, B., Malcolm, J., Stajano, F., Anderson, J., Bonneau, J. (eds.) Security Protocols 2013. LNCS, vol. 8263, pp. 39–44. Springer, Heidelberg (2013)
2. Feamster, N., Rexford, J., Zegura, E.: The road to SDN: an intellectual history of programmable networks. In: Queue, vol. 11, no. 12, pp. 20–32, December 2013
3. Limoncelli, T.: OpenFlow: a radical new idea in networking. In: Queue, vol. 10, no. 6, pp. 40–46, June 2012
4. Caesar, M., Caldwell, D., Feamster, N., Rexford, J., Shaikh, A., van der Merwe, J.: Design and implementation of a routing control platform. In: NSDI 05, pp 15–28 (2005)
5. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
6. Jain, S., Kumar, A., Mandal, S., Ong, J., Poutievski, L., Singh, A., Venkata, S., Wanderer, J., Zhou, J., Zhu, M., Zolla, J., Hölzle, U., Stuart, S., Vahdat, A.: B4: experience with a globally-deployed software defined WAN. In: SIGCOMM 2013. http://cseweb.ucsd.edu/~vahdat/papers/b4-sigcomm13.pdf
7. Cheshire, S.: Latency and the quest for interactivity. http://www.stuartcheshire.org/papers/LatencyQuest.html
8. Geelhoed, E., Parker, A., Williams, D., Groen, M.: Effects of latency on telepresence. Hewlett Packard technical report 120, June 2009. http://www.hpl.hp.com/techreports/2009/HPL-2009-120.pdf
9. Gupta, A., Shahbaz, M., Vanbever, L., Kim, H., Clark, R., Feamster, N., Rexford, J., Shenker, S.: SDX: a software defined internet exchange. Georgia Institute of Technology, SCS technical report; GT-CS-13-06 (2013). https://smartech.gatech.edu/handle/1853/49629
10. Hall, C.: quagga.euro-ix. https://github.com/GMCH