An End to the Middle

Colin Dixon Arvind Krishnamurthy Thomas Anderson
University of Washington

Abstract

The last fifteen years has seen a vast proliferation of middleboxes to solve all manner of persistent limitations in the Internet protocol suite. Examples include firewalls, NATs, load balancers, traffic shapers, deep packet intrusion detection, virtual private networks, network monitors, transparent web caches, content delivery networks, and the list goes on and on. However, most smaller networks in homes, small businesses and the developing world are left without this level of support. Further, the management burden and limitations of middleboxes are apparent even in enterprise networks.

We argue for a shift from using proprietary middlebox harware as the dominant tool for managing networks toward using open software running on end hosts. We show that functionality that seemingly must be in the network, such as NATs and traffic prioritization, can be more cheaply, flexibly, and securely provided by distributed software running on end hosts, working in concert with vastly simplified physical network hardware.

1 Introduction

Middleboxes have done wonderful things for networks in the past few decades. They have enabled us to deal with address-space depletion (NATs), overcome TCP's limitations (packet shapers), and provide better security without changing commodity operating systems (firewalls). Beyond these we rely on virtual private networks (VPNs), proxies, caches, intrusion detection systems (IDSs), load balancers, and many other offerings to keep our networks working properly so we can accomplish our day-to-day tasks.

However, middleboxes have not solved everyone's problems. Most smaller networks have been left out in the cold as the costs to buy and run middlboxes are simply too high. While we could hope that commodity home routers will eventually include all the functionality of the middleboxes we use in enterprise networks, we'd rather not wait. A key motivation for the authors is to be able to get the level of IT support we have at work, for our own networks at home and for networks we help manage for non-profits in the developing world.

We aim to solve two key shortcomings of the current middlebox-based approach to network management in order to make it more effective and its benefits more accessible.

- **Cost** Middleboxes are expensive at almost every level: upfront costs of hardware, skilled staff to manage them, long-term costs of vendor lock-in, scaling to handle increased loads, and so on. In constrast, most computers today are massively overprovisioned, with a proliferating number of cores used only for short bursts of user activity. It should seem strange then to intentionally design a system that keeps functionality in the network and away from endpoints.
- **Perimeter-based** Middleboxes are inherently point solutions, or at best perimeter solutions, yet today's LAN's have increasingly complex internal structure and the proliferation of mobile devices makes the very notion of inside and outside quaint. To get the full benefit of a middlebox, it needs to be deployed throughout the network, not just at the edge. While we can conceivably integrate the full set of middlebox functionality with every router and LAN switch, the result would be neither clean nor cheap.

In the face of these challenges, we propose a more radical, quicker and cheaper option. Specifically, let us consider if we need middleboxes at all. Can we return to a world where we have a simple network with intelligent endpoints and be better off for it? Knowing what we know today, if we were to start from scratch we would not architect non-interoperable complexity into network devices. Rather, we would ask what is the cleanest network we can design to provide us the services we need at reasonable cost?

Several technology trends make this approach increasingly feasible to consider. Perhaps foremost is trusted computing: cheap hardware to validate that a particular device is running a well-known piece of software. In the current model, network administrators operate without the aid of endpoints as they are considered untrusted components. Instead, trusted computing hardware can make it possible to move enforcement onto the endpoint without compromising security. (Of course, some provision must be made in the network for untrusted nodes, but these can be the rare case.) Virtual machines allow this trusted enforcement code to be isolated from the rest of the endpoint operating system; to the user, their computer appears just as configurable as it always has been. Finally, the transition to multicore architectures suggests that all this can be done without even much of a performance penalty, as the PC core running the network enforcement code would likely have been idle in any event.

We call our approach "End to the Middle" or ETTM—

of course there is still a "middle", but it is no longer in control, instead providing a set of simple primitives which we can control from the edge in a similar way to how an operating system controls hardware via the hardware abstraction layer. We want to shift the current practice of proprietary vendor-specific hardware solutions to open source software solutions.

A natural objection to our approach is that we are taking a simple, centralized solution and replacing it with a complex, distributed one. However, the reality is strikingly different. The current solution is far from simple. Rather, in order to provide fault-tolerant and pervasive policy enforcement across a complex network we necessarily must coordinate multiple machines; point solutions only appear simple because they are partial.

Of course, we are not the first to take an operating systems approach to network management. The 4D, CONMan, RCP, NOX, and Maestro projects (8; 3; 4; 10; 5) all attempt to provide a logically centralized management layer on top of a network of diverse routers, switches and middleboxes. In OS terms, this is the equivalent of building a hardware abstraction layer for the network. What differentiates our work from these earlier systems is that we do not treat the network as stopping at the edge of the desktop. Rather, by securely enlisting endpoints, we automatically scale management resources as new load is added into the system. We can also use standard distributed systems techniques to provide fault tolerance in the face of unreliable end hosts. Finally and potentially most important: by placing the majority of network management on commodity hardware running on a well-known, open platform (perhaps even a commodity OS), we can build a much more extensible and simpler system than we could if we have to interoperate with the existing plethora of complex network devices.

The remainder of this paper is organized as follows. Section 2 lays out a high-level architecture of the ETTM approach. We discuss how a few sample network management services might be built on top of that architecture in Section 3. Lastly, we discuss related work in Section 4 and conclude in Section 5.

2 Architecture

2.1 Trust Domains

We focus ETTM on managing the resources of a single organization. This means that in our vision of an ETTM network, all the resources are owned by a single group and we are providing an effective, inexpensive way of giving them control.

Today, the control an organization has largely follows physical boundaries. In general, while the software running on hosts may be mandated to follow certain guidelines, these guidelines are often difficult to enforce and most networks must provide some way for "guest" computers to gain access. Thus, hosts are configured flexibly allowing users to use their machines effective. The network hardware on the other hand is run in a much more tightly controlled fashion. It provides the only effective means of managing the organization's network resources.

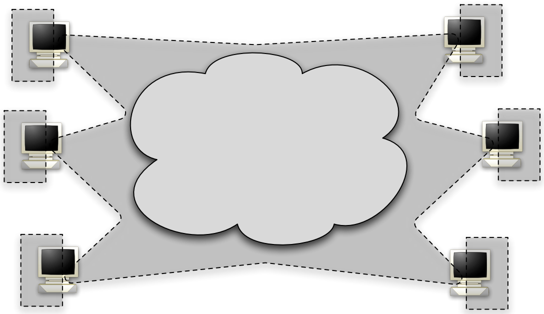

Figure 1: Trust domains in ETTM. Shaded regions indicate trust domains.

Instead, trusted computing enables us to bring end hosts into the fold by verifying that they are running a particular version of the network protocol stack, e.g. on a separate core dedicated to that task. This makes it possible for the network to extend trust beyond the middle as shown in Figure 1. Expanding the network trust domain provides the pervasive deployment needed for effective network policies (effectively having a general-purpose middlebox in front of every machine) as well as the flexibility to optimize, to shift responsibilities to where they can be implemented most efficiently. We will argue that this flexibility can lead to shifting the more complex tasks to the edge leaving a simpler, lower-cost middle.

2.2 Attested Execution Environment

In order to trust end hosts, we assume the presence of a trusted platform module (TPM) and build up from there. By end hosts, we mean desktops, laptops, smart phones, PDA's, and so forth. Many of these devices have TPM support today; we discuss how we accommodate non-TPM devices in the next sub-section. Figure 2 illustrates how we use TPM to secure a portion of the end host for securely managing the network. At boot, a specially-configured hypervisor is loaded which makes two promises. First, it creates a separate network management VM and protects it from any other loaded VMs. We call this VM the Attested Execution Environment (AEE) and all network management tasks occur within it. Second, they hypervisor promises that all outgoing traffic from other virtual machines as well as all incoming traffic will first be routed through the AEE.

The AEE can enforce policy by filtering, shaping, de-

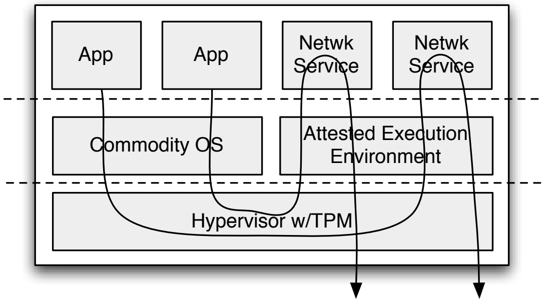

Figure 2: The basic architecture of an ETTM end host.

laying or otherwise handling all traffic sent from or received by the host. It can also execute network management tasks which are less directly related to this host's traffic, for example, to redirect wireless traffic to a mobile host. The network of AEE's can also store and modify network configuration state as well as reconfigure the network, removing the need for a separate network management box. In essence, the set of AEE's becomes the "device driver" for the local area network.

Trusted computing has a negative reputation (2) as its most publicized uses have been to impose restrictions on how people can use their own computers. Just to be clear, this is not what we are trying to do here. We target networks which form a single administrative domain where there is one person or group of people who "own" all of the machines and it is these people who decide what runs. In home networks, one person likely does own all of the computers and so ETTM simply offers more control to manage them. In corporate networks, IT staff already regulate the software which is allowed on computers and visitors can be handled as the rare case.

In addition to providing a trusted environment to safely run code, the AEE also provides a common platform for management tools. Because this platform is run as a VM, it can remain constant across all end hosts providing a standardized tool interface.

2.3 Physical Switches

The goal of physical switches in ETTM is to be simple and uniform in the features they provide. This is in sharp contrast to network hardware vendors who seek ever-increasing functionality without standardization in part to drive vendor lock-in. While it may seem presumptuous to require new kinds of hardware to go in the middle, current projects including OpenWrt (1), OpenFlow (14) and NetFPGA (17) are already building inexpensive, programmable middleboxes which suit our purposes and in the case of OpenFlow the features are often available on existing switches with only a firmware upgrade. These devices are here and more will be coming, it is time we put them to good use.

We envision these devices providing the following features:

- **Neighbor Discovery** To build the network's topology each physical element must be able to identify (and send a packet to) its neighbors. Although end-hosts could ascertain the capacity and latency of links between physical elements without assistance from the network, it is simpler and more accurate for the switches themselves to discover and distribute this information.
- **Switching/Routing** Elements must have programmable lookup tables to determine where to forward each arriving packet, as in OpenFlow (14). Several Ethernet switches now support OpenFlow processing at full link data rate (up to 10GigE). The lookup rules will be dynamically configured by higher-level management tools running on end hosts. If rules become too complex to implement in hardware, source routing can be used as a fallback.
- **Authentication** Physical elements need to be able to establish if a given packet originated from an attested correct software stack or not. On first communication, this requires validating the end host, but subsequently in the common case, most data packets are authenticated by virtue of arriving through a secure channel. Traffic from unauthenticated hosts (e.g. visitors, or machines incapable of supporting an AEE like printers and smartphones) must be redirected to an authenticated management node.
- **Querying** Each element should support end hosts asking questions about its current neighbors and configuration as well as various statistics about its recent and current use and load.

There is one noteworthy omission here: we require no resource allocation or fairness mechanisms but instead rely on the software stacks at end hosts to avoid overdriving network links as well as provide long-term fairness. Physical elements need only provide FIFO queuing and authenticate traffic.

In the common case, end hosts will optimistically assume there is no congestion and simply allow FIFO queuing to take precedence, as queues build up, the extra delay will be noticed and resource allocations for congested areas can be re-negotiated. In fact, some wireless research calls for exactly this kind of end host involvement in top-down bandwidth management. Wireless MACs have tended to inadequately allow for flexible allocation bandwidth leaving few other options.

2.4 Network Management via Distributed Services

On top of the attested execution environment, there are a set of distributed services providing fault-tolerant and pervasive versions of middleboxes, including NATs, DHCP, and the like. To make it easier to build these services, we envision a layer of common building blocks, which we describe next.

2.4.1 Policy Handles

When specifying network policies, we do not want to talk about ports, addresses, and packets, but instead we think in terms of applications and other high-level concepts. For example, an administrator might set mail to have priority over bittorrent downloads; weekly video teleconferences should not be disrupted by software updates. In networks with limited IT budgets, these policies are nearly impossible to implement even with widespread deployment of middleboxes in enterprise networks, these policies are difficult to configure. It is the responsibility of the network management tools to map these high-level concepts into specific low-level enforcement policies.

2.4.2 Resource Discovery & Monitoring

Together the hosts can put together a view of the network including the topology of the network, capacity and latency of links, as well as recent estimates of available bandwidth, and queue lengths. This can be gathered by using active and passive measurements as well as directly querying physical elements. This network view can be used to inform applications, other network management tools, as well as to potentially trigger other actions.

2.4.3 Consensus/Agreement

A crucial component of distributing network management is the ability to agree on the state of the network and the management actions to take. Providing such an agreement protocol as a basic service is valuable; in fact, many middleboxes have started building state replication, failure detection, and consensus into their systems to avoid architecting a single point of failure into their systems.

Consensus algorithms work well in the general case, but dealing with churn and failures can be difficult, especially when failures are correlated (e.g. when a whole group of machines loses power). As long as a majority of machines are up, membership changes can be made to cope with the churn, but we must be able to handle catastrophic failures where a majority of machines might fail simultaneously. In these cases we fall back to the currently communicating nodes declaring a membership change by fiat and deal with merging potential configuration conflicts later. Fortunately, due to the constrained environment of a single organization, we can make stronger assumptions than usual in distributed systems, providing certain options for providing more reliable failure and partition detectors.

Related to consensus is the ability to make a change atomically throughout the network. Most changes will affect more than just one machine or physical element. A partially updated network is unlikely to behave according to either the previous policies or the new ones, but instead in an unpredictable way. To address this issue, it must be possible to make updates atomically across an entire network.

Each different configuration must also have an associated epoch number and each packet should have an epoch number to ensure that it is treated consistently as it crosses the network. After the installation is complete, the old configuration can be thrown away and the old epoch garbage collected.

3 Sample Network Services

3.1 NATs

Network Address Translators (NATs) are some of the most used and vilified middleboxes today. They also provide a reasonable challenge for a system which aims to reduce complexity in the middle as they perform stateful routing.

Our solution is to keep the logical address translation tables at each end host in the network (or possibly a subset for scalability purposes) and ensure it is is consistent using a distributed agreement protocol. At this point each node knows which packets are destined for it and, in fact, knows which node any given packet is destined for.

At the edge of the network, rather than having to keep state and look up where to deliver incoming packets, we can simply forward any given packet to any live host. That host can then forward the packet to its correct destination. We can keep a cache of these mappings as a performance optimization, but this is only soft state. This approach trivially allows for there to be multiple ingress points to the NATed network as well as fault tolerance because no hard state is stored in the network.

Further, tunneling through NATs is vastly simplified as each end host can reconfigure the NAT state whenever `bind()` or `listen()` is called on a socket to provide a globally accessible port.

3.2 Quality of Service

A key advantage of having a foothold on end hosts is that there is a wealth of information available about user intent. Middleboxes, by contrast, must extract this information from network traffic using heuristics that can be foiled by end system changes such as encryption. This is perhaps most important for Quality of Service. As noted in recent work on Network Exception Handlers (12), only the end hosts know which traffic has what importance.

Differentiating between and prioritizing web-browsing, live video conferencing, software updates and e-mail all flowing over port 80 is challenging to impossible from within the network, but if we can look into the operating system and observe which applications are generating or requesting the traffic this task becomes more tractable. Even more advanced quality of service tasks such as providing a boost to traffic related to the currently active window are possible.

4 Related Work

A wealth of recent work has focused on relieving the burden of network management by logically centralizing management tasks. 4D (15; 8; 18), NOX (10), Ethane (7; 6), Network Exception Handlers (12) and CONMan (3) all propose provide some centralized interface to configure a distributed and potentially diverse set of network elements. We continue with the idea of logically centralized management, and we add the novel contribution of using end hosts to implement network management.

Other work has attempted to simplify the networks themselves. OpenFlow (14) looks to add a small degree of programmability to existing switches enabling richer network policies without placing undue burden on the switches themselves. SEATTLE (13) instead aims to remove the need for routers by allowing Ethernet-switching to scale to whole enterprise networks.

Middleboxes have always been a contentious topic, but recent work has looked at how to embrace middleboxes and treat them as first-class citizens. In TRIAD (9) middleboxes are first-order constructs in providing a content-addressable network architecture. The Delegation-Oriented Architecture (16) allows hosts to explicitly invoke middleboxes, while NUTSS (11) proposes a novel connection establishment mechanism which includes negotiation of which middleboxes should be involved.

5 Conclusion

In today's world we want networks that just work. With the help of expensive middleboxes and large, well-trained IT departments, many large enterprise networks come close, but for small networks the story is less positive. For instance, our home networks do not protect web traffic from being delayed in long queues of file-sharing traffic, NATs interfere with peer-to-peer applications and as if that wasn't enough, even simple changes, like adding a second wireless access point to a house involve complex reconfiguration of many different devices. This is far from ideal.

Rather than require a vast array of expensive middleboxes and the associated management overhead, neither of which most small networks can likely afford, we have proposed a different tack. We presented a network architecture which leverages existing resources—namely, end hosts—to provide a network that just works based on shifting management toward the edge.

While we have designed this approach with small networks in mind, we believe that many aspects of the approach may be applicable in larger networks as well. Moving forward, we hope to put this approach into practice in home networks as well as extend the work to handle more diverse networks and management tasks.

References

[1] OpenWrt. `http://openwrt.org/`.

[2] Anderson, R. 'Trusted Computing' Frequently Asked Questions. `http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html`, August 2003.

[3] Ballani, H., and Francis, P. CONMan: A step towards network manageability. In *SIGCOMM* (2007).

[4] Caesar, M., Caldwell, D., Feamster, N., Rexford, J., Shaikh, A., and van der Merwe, J. Design and implementation of a routing control platform. In *NSDI* (2005).

[5] Cai, Z., Cox, A. L., and Ng, T. S. E. Maestro: A new architecture for realizing and managing network controls. In *LISA Workshop on Network Configuration* (2007).

[6] Casado, M., Freedman, M. J., Pettit, J., Luo, J., McKeown, N., and Shenker, S. Ethane: Taking control of the enterprise. In *SIGCOMM* (2007).

[7] Casado, M., Garfinkel, T., Akella, A., Freedman, M. J., Boneh, D., McKeown, N., and Shenker, S. SANE: A protection architecture for enterprise networks. In *USENIX Security* (2006).

[8] 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. In *CCR* (2005).

[9] Gritter, M., and Cheriton, D. R. An architecture for content routing support in the internet. In *USITS* (2001).

[10] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., McKeown, N., and Shenker, S. NOX: Towards an operating system for networks. In *CCR* (2008).

[11] Guha, S., and Francis, P. An end-middle-end approach to connection establishment. In *SIGCOMM* (2007).

[12] Karagiannis, T., Mortier, R., and Rowstron, A. Network exception handlers: Host-network control in enterprise networks. In *SIGCOMM* (2008).

[13] Kim, C., Caesar, M., and Rexford, J. Floodless in SEATTLE: A scalable ethernet architecture for large enterprises. In *SIGCOMM* (2008).

[14] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. OpenFlow: Enabling innovation in campus networks. `http://www.openflowswitch.org/documents/openflow-wp-latest.pdf`, March 2008.

[15] Rexford, J., Greenberg, A., Hjalmtysson, G., Maltz, D. A., Myers, A., Xie, G., Zhan, J., and Zhang, H. Network-wide decision making: Toward a wafer-thin control plane. In *HotNets* (2004).

[16] Walfish, M., Stribling, J., Krohn, M., Balakrishnan, H., Morris, R., and Shenker, S. Middleboxes no longer considered harmful. In *OSDI* (2004).

[17] Watson, G., McKeown, N., and Casado, M. NetFPGA: a tool for network research and education. In *WARFP* (2006).

[18] Yan, H., Maltz, D. A., Ng, T. S. E., Gogineni, H., Zhang, H., and Cai, Z. Tesseract: A 4D network control plane. In *NSDI* (2007).