Middlebox Processing as a Cloud Service

Justine Sherry

Today's networks perform a wide range of specialized tasks such as rewriting packet contents to make use of new protocols or scanning packets for evidence of malicious activity. These tasks are performed by special-purpose devices called middleboxes. While once considered a rarity, my research has shown that middleboxes are ubiquitously deployed and yet costly and complicated to manage. My dissertation advocates an alternate architecture for supporting middlebox functionality: I propose that middlebox functionality be implemented as a software service that runs in the cloud. I have designed, implemented and evaluated systems that allow enterprises to outsource middlebox processing to third party providers, demonstrating not only how to implement outsourcing, but also how to provide fault-tolerance and privacy for software-based middleboxes. Thus, my research brings the benefits of cloud computing to networking. Some of the technologies I have developed are already being adopted in industrial systems.

Overview

The Internet was originally designed with one service as its goal: best-effort packet delivery. Networks were composed of routers and switches whose sole task was to read each packet's destination and decide where to send it. Today, the Internet has grown to do a tremendous amount more. In 2012, I performed a survey [1, 2] which showed that roughly one in three devices in an enterprise network is a *middlebox*: a special-purpose device that may inspect, transform, or modify packets to improve performance and security. A middlebox may scan a connection for malicious behavior, compress data to provide better performance on low-resource mobile devices, or serve content from a local cache to reduce bandwidth costs.

My survey was surprising to the networking community; where many thought that middleboxes were rare, special case devices, my survey showed that they were deployed near-ubiquitously. At the time of this writing the survey is cited over 200 times on Google Scholar by follow-on work exploring how middleboxes change assumptions in wide-ranging topics such as software defined networking, network management, Internet architecture, protocol design, and network measurement.

My dissertation research follows from my survey and proposes a radical change in how middleboxes are implemented and deployed. In my survey, administrators reported that middleboxes were complex, costly, often the cause of downtime, and that misconfiguration of the devices was common. To solve all of these challenges at once, I proposed that enterprise network administrators outsource middlebox processing to third party services hosted by cloud providers or ISPs. Service providers would implement middleboxes using software virtualization in a similar way to how they offer compute and storage services. For enterprise networks, outsourcing processing promises to reduce costs, ease management, and provide resources for scalability and failover. In effect, this shift would bring the benefits of cloud computing to networking infrastructure, moving all but the most low-level functionality out of enterprises and to the cloud. Few modern enterprises operate their own mail or storage server any more, and I think network services should be no different.

To understand the performance impact and benefits of outsourcing at potential clients, I designed, implemented, and deployed a system called APLOMB [1] on EC2. Using APLOMB and a case study of a multinational enterprise, I concluded that outsourcing of network processing would allow an enterprise to outsource over 90% of its middleboxes and would have modest performance overheads.

While APLOMB showed that outsourcing would be feasible and beneficial for *clients*, it far from closed the story on how a *cloud provider* should implement and offer middlebox functionality in the cloud. Two obstacles to providing competitive and desirable service in clouds are fault-tolerance and privacy. Many engineering hours have been devoted to both of these problems for traditional cloud systems. Nonetheless, existing solutions are unprepared to support middlebox workloads because middlebox performance requirements are more demanding than other cloud systems. For example, a cluster compute task using Spark might complete in seconds or a web server might load

a page in tens of milliseconds, but a middlebox will process a packet in under 100 microseconds. Twitter's stream processing system requires 500 cores to process up to 20 million tuples per second; a *single network link* of 10Gbps can generate 14.8 million packets per second.

My dissertation presents new solutions to classical cloud challenges, rethinking them from the ground up to achieve the tight performance requirements in packet processing environments. FTMB [3] is a system which provides fault-tolerance for middlebox applications while increasing per packet latencies by only 30μs – 2-3 orders of magnitude lower than approaches for general cloud computing. To provide privacy guarantees, BlindBox [4] and MBArk [5] allow a cloud provider to perform certain middlebox operations over encrypted connections. BlindBox and MBArk present new functional cryptography techniques tuned to packet processing to sustain good throughput, also as much as 3-5 orders of magnitude faster than functional cryptography solutions designed for the cloud space.

Approach. My research approach involves a tight loop of iteration between practitioners, implementation, and research. Correspondingly, my first step in exploring the middlebox problem space was to survey 57 network administrators about their day-to-day experiences deploying and managing middleboxes. Through these conversations I was able to identify real and relevant problems (complexity, failures) in need of a solution. In solving a new research challenge, I look broadly across the toolkit of computer science – often learning new skills or creating new collaborations to solve the problem at hand. This approach led me to learn elementary network calculus (for Silo [6]), dive in to distributed systems (for FTMB [3]), and explore the capabilities of functional cryptography (for BlindBox [4] and MBArk [5]). Where some researchers specialize in a particular 'hammer' and go searching for 'nails', I focus on the nail first and then develop the appropriate hammer.

Impact. APLOMB in some ways anticipated a new industry movement called Network Functions Virtualization (NFV). NFV was proposed mid-way through my PhD by ETSI, a consortium of Internet Service Providers and network infrastructure vendors. NFV aims to move middlebox packet processing from dedicated, special purpose hardware on to general-purpose infrastructure using software and virtualization – just as I proposed that cloud providers do in APLOMB. With NFV, I have found allies in pursuing my research agenda. For example, AT&T has adopted FTMB [3] and submitted it to ETSI [7], which today operates as a standards body for NFV.

Overall, I believe that software packet processing, middleboxes, and the emergence of NFV make it an exciting time to be in networking. Today, we are increasingly seeing clouds and ISPs inserting generic compute capabilities and software services on the dataplane. With middleboxes running *in software* on *public infrastructure*, we have a new platform upon which new ideas in protocol design, packet scheduling, network services, security infrastructure, and other features can be easily implemented and deployed. Years from now I hope to look back on many research projects across different areas; however my students will be able to *deploy* their new ideas on ISP and cloud infrastructure just as easily as startups and research deploy code to EC2 and Azure.

In what follows, I will describe my dissertation research on middleboxes as a cloud service, discuss other research I have done in networking, and finally sketch some thoughts on future research directions.

Thesis Work: Middlebox Processing as a Cloud Service

I will now describe the systems that compose my thesis in detail.

APLOMB: Middlebox Outsourcing

The Problem. My survey showed that middleboxes increased complexity and cost for enterprises. For example, a given enterprise might deploy as many as 8 types of middleboxes. Large companies would deploy hundreds of middleboxes, and a cycle of three year upgrades meant that administrators were constantly learning how to manage a new class of device. Each device could cost tens or hundreds of thousands of dollars, and each deployment required large operational teams (with salaries reflecting their expertise) to manage the devices.

The Solution. APLOMB [1], which appeared in SIGCOMM 2012, presents an architecture based on two insights. First, the average enterprise should not be concerned with this complexity – and should instead *outsource* network processing the same way that compute and storage services are often outsourced today. Second, middlebox processing should be managed by a cloud provider in software as a virtualized service.

The Challenge. For outsourcing to be feasible, we needed to know three things. First, that it could be done simply and cheaply – if implementing traffic redirection to and from the cloud were more complicated than deploying middleboxes in the enterprise, it would defeat the purpose altogether. Second, we needed to know that we could achieve functional

equivalence: that the benefits of middleboxes would not be lost by relocating them from enterprise to cloud. Finally, we needed to be sure that the performance overheads of redirecting traffic to and from a cloud provider would not prohibit practical usage.

I designed, implemented, and deployed APLOMB and hence was able to show by existence proof that outsourcing could be performed simply, achieve almost all of the benefits as standard middlebox deployments, and with low performance overheads. Combining the APLOMB deployment with a case study of a multinational enterprise showed that APLOMB would be beneficial in practice.

Since Publication. Outsourcing traffic processing has emerged as an offering from cloud-based startups such as Aryaka for WAN Optimization and ZScalar for intrusion detection. Furthermore, AT&T has sketched a vision to implement middlebox services on their own infrastructure, which they refer to as Domain 2.0.

FTMB: Middlebox Fault Tolerance

The Problem. Middleboxes are stateful systems. Losing middlebox state can result in loss of connectivity, lost income, or missed attacks. Since cloud providers often differentiate themselves based on guaranteed uptimes and SLAs, I believed that fault tolerance solutions were a key component for a service provider to offer a competitive and desirable service. Middlebox vendors focused on solutions based on fixed-function hardware devices that they themselves would provide, but in moving to the cloud we needed a general solution based entirely in software. Many software-based cloud solutions to fault-tolerance exist for, e.g., cluster compute frameworks or web services, but none could cope with the performance requirements of middleboxes. General-purpose fault-tolerance systems we tested increased per-packet latencies by 8-50ms. For many applications these overheads are considered negligible, but in packet processing – where page download times increase by a multiplicative factor relative to latency, and companies invest in shaving off every last millisecond – these overheads were unacceptable.

The Solution. FTMB [3], which appeared in SIGCOMM 2015 and was awarded Best Student Paper, implements a software based recovery solution whose roots go back to distributed systems. FTMB achieves a median latency overhead of only 30μs per packet – 2-3 orders of magnitude lower than software fault tolerance approaches for general cloud systems.

The Challenge. All fault-tolerance systems must address a property called 'output commit' – whenever releasing data to a user, the system guarantees its ability to recall all state relevant to that data (and that if the master system goes down, a backup or replica will remember the state too). In databases, this is equivalent to 'durability' for transactions. At 10Gbps, a packet is released *every microsecond*, meaning the system must 'commit' state on similar timescales. FTMB achieves good performance by optimizing this ability to commit data quickly in packet processors.

Since Publication. Outside of academia, FTMB has been published by the European Telecommunication Standards Instituted (ETSI) in an informational draft proposed by AT&T [7]. In addition, two major middlebox vendors have FTMB in trials on their products. I am currently collaborating with these vendors to understand how FTMB performs in their systems, and we plan to produce an 'industrial systems' report on FTMB in practice.

BlindBox and MBArk: Middlebox Privacy

The Problem. Privacy is a constant concern in migrating to cloud services, and packet processing is no exception. Indeed, while outsourcing storage to a cloud provider exposes the stored data to the cloud provider, a user who outsources traffic processing to the cloud grants the cloud provider access to *all data they transmit over the Internet to any other host.*

The Solution. Outsourcing and privacy have traditionally seemed fundamentally at odds with each other, but I learned this was untrue when I attended a talk by a faculty candidate at Berkeley, Raluca Ada Popa. During her talk, I realized that functional encryption could be used to allow middleboxes to operate over connection data *while the data remains encrypted.* Shortly after, I approached her with my idea and asked her to teach me about functional cryptography so that I could apply it to packet processing. Out of our collaboration came two systems: BlindBox [4], which appeared in SIGCOMM 2015, and MBArk [5], which will appear in NSDI 2016.

The Challenge. Once again, performance was the key obstacle to this project. Functional cryptography has only recently been made practical and even then only applied to systems like databases and web servers. Middleboxes once again pushed the limits of what performance was possible. BlindBox focused on one particular application, signature-based intrusion detection systems, which rely on detecting when known-malicious substrings of data (e.g., a snippet of malicious javascript) appear in a bytestream. BlindBox enables a middlebox to detect malicious strings

over data that remains encrypted using a new searchable encryption algorithm that is 3-6 orders of magnitude faster at packet processing than existing schemes from other systems. Nonetheless, BlindBox was only a first step in the space, and suffered poor performance and a lack of generality. MBArk, led by Chang Lan (another student in my group), generalized to many other middleboxes and improved many of BlindBox's performance problems.

Since Publication. This work opens more questions than it answers. MBArk removed many of BlindBox's overheads, but continues to have high bandwidth overheads: one open research challenge is how to reduce this. Another line of questioning involves how to design rulesets for MBArk and BlindBox: how can we ensure that the substrings in a ruleset are 'safe' to use and only detect attacks, and do not leak information a user might wish to keep private? Where APLOMB and FTMB resulted in systems 'ready for deployment', I consider this set of work more forward-looking.

Other Research

Outside of my dissertation work, I have also explored topics in network congestion control, datacenter performance and in Internet measurement. While my dissertation focused on middlebox processing, I consider myself broadly interested in networks and enjoyed the opportunity to explore these other problem domains.

RC3: Recursively Cautions Congestion Control [NSDI 2014, HotNets 2013] [8, 9] When an end host opens a new connection, it must decide how fast to send. If a user sends at too a high rate initially, it interferes with other users, causing packet loss, and poor performance. TCP therefore opts for the cautious route but consequently starts transmitting more slowly than necessary. RC3, our proposed protocol, uses low priority traffic to allow a sender to transmit at maximum rate at startup without interfering with other users. Overall, we showed that RC3 can improve flow completion times over the Internet on average by 40%, with improvements of up to 70% for medium to large flows.

Silo: Predictable Message Latency in the Cloud [SIGCOMM 2015] [6] In multitenant clouds like Azure and EC2, networking is a shared resource, and latency and throughput between virtual machines can vary widely depending on where virtual machines are placed and on utilization demands of neighboring virtual machines. We designed Silo to give clients bandwidth *and latency* guarantees, thus improving application performance and predictability.

IP Alias Resolution with Prespecified Timestamps [IMC 2010] [10] Topology measurements are often used by operators to diagnose network outages and failures, especially when outages in one network impact connectivity in another. We showed how to use IP timestamps, a little known extension to IP, to improve the correctness and coverage of topology measurements. The Center for Applied Internet Data Analysis (CAIDA) built a tool called Motu [11] based on my techniques.

Reverse Traceroute [NSDI 2010] [12] Traceroute is the go-to tool for diagnosing failures on the Internet. Traceroute shows an operator the path *from* their computer *to* any destination on the Internet. However, a longstanding challenge was how to measure the path *from* another server *to* an administrator's local computer without having to gain control of the remote server. Reverse traceroute uses an Internet-wide system of vantage points to measure reverse paths, and was awarded Best Paper at NSDI 2010.

Future Work

Currently, there is a tremendous amount of momentum behind software packet processing. I would like to see FTMB through to commercial deployment, and there remain several challenges and obstacles to migrating middleboxes fully to virtualized software deployments, including the following:

Scale-Out Appliances. Middleboxes currently scale *up* to some extent via multicore, fast processors, and large memory, but they don't scale *out* well. The scale-out problem is challenging because some middleboxes keep data that must be analyzed in aggregate across multiple flows or users (e.g., the set of all connections initiated due to loading a particular URL or all hosts in a network running a particular operating system) that require coordination across multiple servers. Cross-node coordination while processing a packet is prohibitively expensive, and hence any scale-out system to scale out must offer relaxed consistency guarantees.

Multitenancy. Middleboxes are currently designed for deployment in enterprises under a single administrative domain. In moving to a model of shared infrastructure offered by an ISP or cloud, the prevailing model is giving each client an independent virtual machine. However, I learned recently while talking to a major cloud provider that this approach has two major shortcomings: it fragments operator control over a single service in to many small services, and

it wastes resources due to virtualization overheads. Is the right path instead to design applications with multitenancy as a first principle, as many shared storage systems are designed?

Human-Intelligibility. The overarching goal of the move to virtualized networking appliances is to lessen complexity for humans. Nonetheless, few research papers actually return to human administrators to evaluate usability. Evaluating new middlebox management initiatives under the lens of HCI seems like a key piece to understanding whether or not this research meets its goals.

In the longer term, my work in software packet processing and my collaborations in other areas have started to pull my interests in to various 'offshoot' directions outside of my core thesis work. One of these offshoots is a growing interest in privacy. I cannot help but note that my work on middleboxes has enabled yet another way to track user behavior on the Internet. Designing BlindBox and MBArk were my first steps in exploring how to regain some user control over what providers learn about them. These two projects gave me greater consciousness about what information a user might wish to keep private that even modern protocols like TLS or IPSec may reveal. Trends like 'online by default' public services and the Internet of Things are only increasing the amount of data vulnerable to leakage.

Overall, I remain broadly interested in systems, networking, and the challenges that come my way in these fields.

References

[1] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, "Making Middleboxes Someone Else's Problem: Network Processing As a Cloud Service," in *Proceedings of the ACM SIGCOMM 2012 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '12, (New York, NY, USA), pp. 13–24, ACM, 2012.

[2] J. Sherry and S. Ratnasamy, "A Survey of Enterprise Middlebox Deployments," Tech. Rep. UCB/EECS-2012-24, EECS Department, University of California, Berkeley, Feb 2012.

[3] J. Sherry, P. X. Gao, S. Basu, A. Panda, A. Krishnamurthy, C. Maciocco, M. Manesh, J. a. Martins, S. Ratnasamy, L. Rizzo, and S. Shenker, "Rollback-Recovery for Middleboxes," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, (New York, NY, USA), pp. 227–240, ACM, 2015.

[4] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy, "BlindBox: Deep Packet Inspection over Encrypted Traffic," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, (New York, NY, USA), pp. 213–226, ACM, 2015.

[5] C. Lan, J. Sherry, R. A. Popa, and S. Ratnasamy, "MBArk: Securely Outsourcing Middlebox Processing to the Cloud," in *Proceedings of the 13th USENIX Conference on Networked Systems Design and Implementation*, NSDI'16, (Berkeley, CA, USA), USENIX Association, 2016.

[6] K. Jang, J. Sherry, H. Ballani, and T. Moncaster, "Silo: Predictable Message Latency in the Cloud," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, (New York, NY, USA), pp. 435–448, ACM, 2015.

[7] P. Tarpore, R. Schlichting, R. Adams, S. Antzen, L. Chidung, G. Singh, P. Vaananen, and C. Lui, "Network Functions Virtualisation (NFV); Reliability; Report on Scalable Architectures for Reliability Management." European Telecommunications Standards Institute. GS NFV-REL 002. https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=44582.

[8] R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "Recursively cautious congestion control," in *Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation*, NSDI'14, (Berkeley, CA, USA), pp. 373–385, USENIX Association, 2014.

[9] R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "How to Improve Your Network Performance by Asking Your Provider for Worse Service," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, HotNets-XII, (New York, NY, USA), pp. 25:1–25:7, ACM, 2013.

[10] J. Sherry, E. Katz-Bassett, M. Pimenova, H. V. Madhyastha, T. Anderson, and A. Krishnamurthy, "Resolving IP Aliases with Prespecified Timestamps," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*, IMC '10, (New York, NY, USA), pp. 172–178, ACM, 2010.

[11] Center for Internet Data Analysis, "Motu Dealiasing Tool." https://www.caida.org/tools/measurement/motu/.

[12] E. Katz-Bassett, H. V. Madhyastha, V. K. Adhikari, C. Scott, J. Sherry, P. Van Wesep, T. Anderson, and A. Krishnamurthy, "Reverse traceroute," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, NSDI'10, (Berkeley, CA, USA), pp. 15–15, USENIX Association, 2010.