Making Middleboxes Someone Else's Problem: Network Processing as a Cloud Service

Justine Sherry* Vyas Sekar† Shaddi Hasan*
Colin Scott* Arvind Krishnamurthy‡ Sylvia Ratnasamy*

ABSTRACT

Modern enterprises almost ubiquitously deploy middlebox processing services to improve security and performance in their networks. Despite this, we find that today's middlebox infrastructure is expensive, complex to manage, and creates new failure modes for the networks that use them. Given the promise of cloud computing to decrease costs, ease management, and provide elasticity and fault-tolerance, we argue that middlebox processing is a natural fit for the cloud. Arriving at a feasible implementation, however, is challenging due to the need to achieve functional equivalence with traditional middlebox deployments without sacrificing performance or increasing network complexity.

In this paper, we motivate, design, and implement APLOMB, a practical service for outsourcing enterprise middlebox processing to the cloud. Our discussion of APLOMB is data-driven, guided by a survey of 57 enterprise networks, the first large-scale academic study of middlebox deployment. We show that APLOMB solves real problems faced by network administrators, can outsource over 90% of middlebox hardware in a typical large enterprise network, and, in a case study of a real enterprise, imposes an average latency penalty of 1.1ms and median bandwidth inflation of 3.8%.

1. INTRODUCTION

Today's enterprise networks rely on a wide spectrum of specialized appliances or *middleboxes*. Trends such as the proliferation of smartphones and wireless video are set to further expand the range of middlebox applications [21]. Middleboxes offer valuable benefits, such as improved security (*e.g.*, firewalls and intrusion detection systems), improved performance (*e.g.*, proxies) and reduced bandwidth costs (*e.g.*, WAN optimizers). However, as we show in §2, middleboxes come with high infrastructure and management costs, which result from their complex and specialized processing, variations in management tools across devices and vendors, and the need to consider policy interactions between these appliance and other network infrastructure.

The above shortcomings mirror the concerns that first motivated enterprises to transition their in-house IT infrastructures to managed cloud services. Inspired by this trend, we ask whether the promised benefits of cloud computing—reduced expenditure for infrastructure, personnel and management, pay-by-use, the flexibility to try new services without sunk costs, *etc.*—can be brought to middlebox infrastructure. Beyond improving the status quo, cloud-based middlebox services would also make the security and performance benefits of middleboxes available to users who cannot otherwise afford the associated costs and complexity—small businesses and home and mobile users.

We envision enterprises outsourcing the processing of their traffic to third-party *middlebox service providers* operating middleboxes in the cloud. Our proposal represents a significant change to enterprise networks, and hence we first validate that this exercise is worthwhile by examining what kind of a burden middleboxes impose on enterprises. The research literature, however, offers surprisingly few studies of middleboxes in real-world enterprises ([44] comes closest with anecdotal reports from a single large enterprise). We thus start with a study of 57 enterprise networks, aimed at understanding (1) the nature of real-world middlebox deployments (*e.g.*, types and numbers of middleboxes), (2) "pain points" for network administrators, and (3) failure modes. Our study reveals that middleboxes do impose significant infrastructure and management overhead across a spectrum of enterprise networks and that the typical amount of middleboxes in an enterprise is comparable to its traditional L2/L3 infrastructure!

Having established the costs associated with middlebox deployments and the potential benefits of outsourcing them, we examine different options for architecting cloud-based middlebox services. To be viable, such an architecture must meet three challenges:

(1) Functional equivalence. A cloud-based middlebox must offer functionality and semantics equivalent to that of an on-site middlebox—*i.e.*, a firewall must drop packets correctly, an intrusion detection system (IDS) must trigger identical alarms, *etc.* In contrast to traditional endpoint applications, this is challenging because middlebox functionality may be topology dependent. For example, traffic compression must be implemented *before* traffic leaves the enterprise access link, and an IDS that requires stateful processing must see *all* packets in *both* directions of a flow. Today, these requirements are met by deliberately placing middleboxes 'on path' and/or at network choke points—options that are not readily available in a cloud-based architecture. As we shall see, these topological constraints fundamentally complicate our ability to outsource middlebox processing.

(2) Low complexity at the enterprise. An outsourced middlebox architecture still requires some supporting functionality

*UC Berkeley,† Intel Research,‡ University of Washington

at the enterprise. We aim for a cloud-based middlebox architecture that minimizes the complexity of this enterprise-side functionality: failing to do so would detract from our motivation for outsourcing in the first place.

(3) Low performance overhead. Middleboxes today are located on the direct path between two communicating endpoints. Under our proposed architecture, traffic is instead sent on a detour through the cloud leading to a potential increase in packet latency and bandwidth consumption. We aim for system designs that minimize this performance penalty.

We explore points in a broad design space that is defined by the redirection options available to enterprises, the footprint of the cloud provider, and the complexity of the outsourcing mechanism. We find that all options have natural tradeoffs across the above requirements and settle on a design, termed APLOMB[1], that we argue is the sweet spot in this design space. We implement APLOMB and evaluate our system on EC2 using real end-user traffic and an analysis of traffic traces from a large enterprise network, where APLOMB imposes an average latency increase of only 1 ms and a median bandwidth inflation of 3.8%.

To summarize, our key contributions are:

- A study of costs and concerns in 57 real-world middlebox deployments, across a range of enterprise scenarios.
- A systematic exploration of the design requirements and design space for outsourcing middleboxes.
- The design, implementation, and evaluation of the APLOMB architecture.
- A case study of how our system would impact the middlebox deployment of a large enterprise.

It is worth noting that a core question in network design is where network functionality should be embedded. A wealth of research has explored this question for various network functionality, such as endpoints *vs.* routers for congestion control [24, 38, 34] and on-path routers *vs.* off-path controllers for routing control plane functions [33, 41]. Our work follows in this vein: the functionality we focus on is advanced traffic processing, an increasingly important piece of the network *data* plane, and we weigh the relative benefits of embedding such processing in the cloud *vs.* on-path middleboxes, under the conjecture that the advent of cloud computing offers new, perhaps better, options for supporting middlebox functionality.

Roadmap: We present our study of enterprise middlebox deployments in §2. §3 explores the design space for outsourcing middleboxes. We present the design and evaluation of the APLOMB architecture in §4 and §5 respectively. We discuss related work in §7 before concluding in §8.

2. MIDDLEBOXES TODAY

Before discussing outsourcing designs, we draw on two datasets to discuss typical middlebox deployments in enterprise networks and why their challenges might be solved by the cloud. We conducted a survey of 57 enterprise network administrators regarding their networks, including the number of middleboxes deployed, personnel dedicated to them, and challenges faced in administering them. To the best of our knowledge, this is the *first large-scale survey* of middlebox deployments in the research community. Our dataset includes 19 small (fewer than 1k hosts) networks, 18 medium (1k-10k hosts) networks, 11 large (10k-100k hosts) networks, and 7 very large (more than 100k hosts) networks. We augment our analysis with measurements from a large enterprise with approximately 600 middleboxes and tens of international sites; we elaborate on this dataset in §5.3.

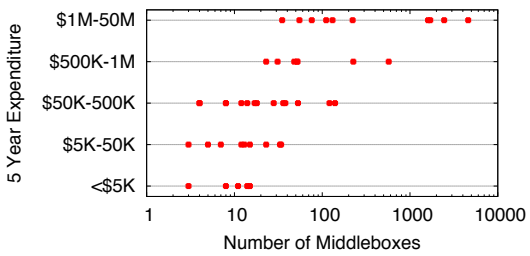


Figure 2: Administrator-estimated spending on middlebox hardware per network.

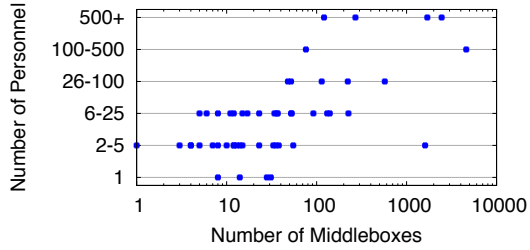


Figure 3: Administrator-estimated number of personnel per network.

Our analysis highlights several key challenges that enterprise administrators face with middlebox deployments: large deployments with high capital expenses and operating costs (§2.1), complex management requirements (§2.2), and the need for overprovisioning to react to failure and overload scenarios (§2.3). We argue these factors parallel common arguments for cloud computation, and thus make middleboxes good candidates for the cloud.

2.1 Middlebox Deployments

Our data illustrates that typical enterprise networks are a complex ecosystem of firewalls, IDSes, web proxies, and other devices. Figure 1 shows a box plot of the number of middleboxes deployed in networks of all sizes, as well as the number of routers and switches for comparison. Across all network sizes, the number of middleboxes is on par with the number of routers in a network! The average very large network in our data set hosts 2850 L3 routers, and 1946 total middleboxes; the average small network in our data set hosts

[1]APLOMB stands for Architecture/Appliance for Outsourcing Middleboxes.

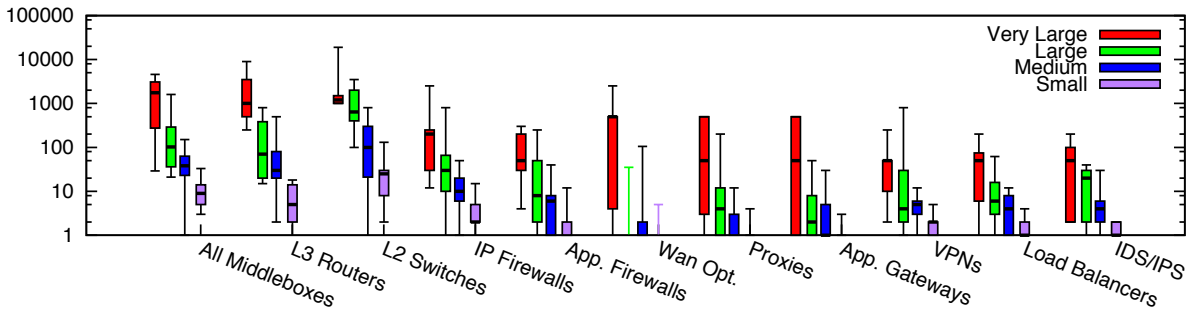


Figure 1: Box plot of middlebox deployments for small (fewer than 1k hosts), medium (1k-10k hosts), large (10k-100k hosts), and very large (more than 100k hosts) enterprise networks. Y-axis is in log scale.

7.3 L3 routers and 10.2 total middleboxes.[2]

These deployments are not only large, but are also costly, requiring high up-front investment in hardware: thousands to millions of dollars in physical equipment. Figure 2 displays five year expenditures on middlebox hardware against the number of actively deployed middleboxes in the network. All of our surveyed very large networks had spent over a million dollars on middlebox hardware in the last five years; the median small network spent between $5,000-50,000 dollars, and the top third of the small networks spent over $50,000.

Paralleling arguments for cloud computing, outsourcing middlebox processing can reduce hardware costs: outsourcing eliminates most of the infrastructure at the enterprise, and a cloud provider can provide the same resources at lower cost due to economies of scale.

2.2 Complexity in Management

Figure 1 also shows that middleboxes deployments are diverse. Of the eight middlebox categories we present in Figure 1, the median very large network deployed seven categories of middleboxes, and the median small network deployed middleboxes from four. Our categories are coarse-grained (*e.g.* Application Gateways include smartphone proxies and VoIP gateways), so these figures represent a *lower bound* on the number of distinct device types in the network.

Managing many heterogeneous devices requires broad expertise and consequently a large management team. Figure 3 correlates the number of middleboxes against the number of networking personnel. Even small networks with only tens of middleboxes typically required a management team of 6-25 personnel.[3] Thus, middlebox deployments incur substantial operational expenses on top of hardware costs.

Understanding the administrative tasks required to manage middleboxes further illuminates why large administrative staffs are needed. We break down the management tasks related to middleboxes as follows:

Upgrades and Vendor Interaction. Deploying new features in the network entails deploying new hardware infrastructure; network operators upgrade, in the median case, every four years. Each time they negotiate a new deployment, they must select between several offerings, weighing the capabilities of devices offered by numerous vendors – an average network in our dataset contracted with 4.9 vendors. This four-year cycle is at the same time both too frequent and too infrequent. Upgrades are too frequent in that every four years, administrators must evaluate, select, purchase, install, and train to maintain new appliances. Upgrades are too infrequent in that administrators are 'locked in' to hardware upgrades to obtain new features. Quoting one administrator:

> Upgradability is very important to me. I do not like it when vendors force me to buy new equipment when a software upgrade could give me additional features.

Cloud computing eliminates the upgrade problem: enterprises sign up for a middlebox *service*; how the cloud provider chooses to upgrade hardware is orthogonal to the service offered.

Monitoring and Diagnostics. To make managing tens or hundreds of devices feasible, enterprises deploy network management tools (e.g., [18, 11]) to aggregate exported monitoring data, *e.g.* SNMP. However, with a cloud solution, the cloud provider monitors utilization and failures of specific devices, and only exposes a middlebox *service* to the enterprise administrators, simplifying management at the enterprise.

Configuration. Configuring middleboxes requires two tasks. *Appliance configuration* includes allocating IP addresses, installing upgrades, and configuring caches. *Policy configuration* is tailoring the device to enforce specific enterprise-wide policy goals (*e.g.* HTTP blocks filters social networks). Cloud-based deployments obviate the need for enterprise administrators to focus on the low-level mechanisms for appliance configuration and focus only on policy configuration.

Training. New appliances require new training for administrators to manage them. One administrator even stated that

[2]Even 7.3 routers and 10.2 middleboxes represents a network of a substantial size. Our data was primarily surveyed from the NANOG network operators group, and thus does not include many of the very smallest networks (*e.g.* homes and very small businesses with only tens of hosts).

[3]We've contacted some of the administrators who reported the strongest outliers in this graph; we expect that most outliers under- or over-reported their personnel counts. Nonetheless, the median cases still represent substantial administrative personnel.

	Misconfig.	Overload	Physical/Electric
Firewalls	67.3%	16.3%	16.3%
Proxies	63.2%	15.7%	21.1%
IDS	54.5%	11.4%	34%

Table 1: Fraction of network administrators who estimated misconfiguration, overload, or physical/electrical failure as the most common cause of middlebox failure.

existing training and expertise was a key question in purchasing decisions:

> Do we have the expertise necessary to use the product, or would we have to invest significant resources to use it?

Another administrator reports that a lack of training limits the benefits from use of middleboxes:

> They [middleboxes] could provide more benefit if there was better management, and allocation of training and lab resources for network devices.

Outsourcing diminishes the training problem by offloading many administrative tasks to the cloud provider, reducing the set of tasks an administrator must be able perform. In summary, for each management task, outsourcing eliminates or greatly simplifies management complexity.

2.3 Overload and Failures

Most administrators who described their role as engineering estimated spending between one and five hours per week dealing with middlebox failures; 9% spent between six and ten hours per week. Table 1 shows the fraction of network administrators who labeled misconfiguration, overload, and physical/electrical failures the most common cause of failures in their deployments of three types of middleboxes. Note that this table is *not* the fraction of failures caused by these issues; it is the fraction of administrators who estimate each issue to be the *most common* cause of failure. A majority of administrators stated misconfiguration as the most common cause of failure; in the previous subsection we highlight management complexity which likely contributes to this figure.

On the other hand, many administrators saw overload and physical/electrical problems as the most common causes of errors. For exampe, roughly 16% of administrators said that overload was the most common cause of IDS and proxy failure, and 20% said that physical failures were the most common cause for proxies. A cloud-based capacity to elastically provision resources avoids overload by enabling on-demand scaling and resolves failure with standby devices – without the need for expensive overprovisioning.

2.4 Discussion

To recap, our survey across 57 enterprises illuminates several middlebox-specific challenges that cloud outsourcing can solve: large deployments with high capital and operating expenses, complex management requirements inflating

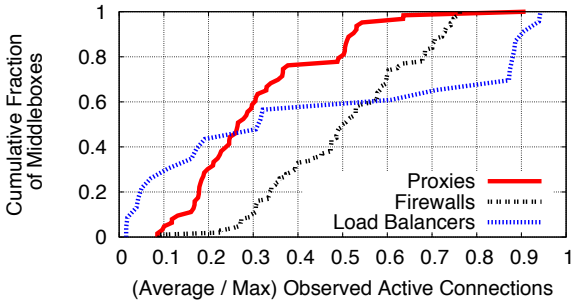


Figure 4: Ratio of average to peak active connections for all proxies, firewalls, and load balancers in the very large enterprise dataset.

operation expenses, and failures from physical infrastructure and overload. Cloud outsourcing can cut costs by leveraging economies of scale, simplify management for enterprise administrators, and can provide elastic scaling to limit failures.

Outsourcing to the cloud not only solves challenges in existing deployments, but also presents new opportunities. For example, resource elasticity not only allows usage to scale *up*, but also to scale *down*. Figure 4 shows the distribution of average-to-max utilization (in terms of active connections) for three devices across one large enterprise. We see that most devices operate at moderate to low utilization; e.g., 20% of Load Balancers run at $<$5% utilization. Today, however, enterprises must invest resources for peak utilization. With a cloud solution, an enterprise can lease a large load balancer only at peak hours and a smaller, cheaper instance otherwise. Furthermore, a pay-per-use model democratizes access to middlebox services and enables even small networks who cannot afford up-front costs to benefit from middlebox processing.

These arguments parallel familiar arguments for the move to cloud computation [27]. This parallel, we believe, only bolsters the case.

3. DESIGN SPACE

Having established the potential benefits of outsourcing middleboxes to the cloud, we now consider how such outsourcing might be achieved. To start, any solution will require some supporting functionality deployed at the enterprise: at a minimum, we will require some device to *redirect* the enterprise's traffic to the cloud. Hence, we assume that each enterprise deploys a generic appliance which we call an *Appliance for Outsourcing Middleboxes* or *APLOMB*. However, depending on the complexity of the design, the functionality might be integrated with the egress router. We assume that the APLOMB redirects traffic to a Point of Presence (PoP), a datacenter hosting middleboxes which process the enterprise's traffic.

As a baseline, we reflect on the properties of middleboxes as deployed today within the enterprise. Consider a middlebox m that serves traffic between endpoints a and b. Our proposal is to change the *placement* of m – moving m from

the enterprise to the cloud. Moving m to the cloud eliminates three key properties of its current placement:
(1) on-path: m lies on the direct IP path between a and b
(2) choke point: all paths between a and b traverse m
(3) local: m is located inside the enterprise.

The challenges we face in outsourcing middleboxes all derive from losing the above properties, and our design focuses on compensating for this loss. More specifically, in attempting to regain the benefits of the above properties, we arrive at three design components, as described below.

Redirection: Being on-path makes it trivially easy for a middlebox to obtain the traffic it must process; being at a choke point ensures the middlebox sees both directions of traffic flow between two endpoints (bidirectional visibility is critical since most middleboxes operate at the session level). A middlebox in the cloud loses this natural ability; hence we need a redirection architecture that routes traffic between a and b via the cloud, with both directions of traffic consistently traversing the same cloud PoP.

Latency Strategy: A second consequence of being on-path is that the middlebox introduces no additional latency into the path. In contrast, sending traffic on a detour through the cloud could increase path latency, necessitating a practical strategy for low latency operation.

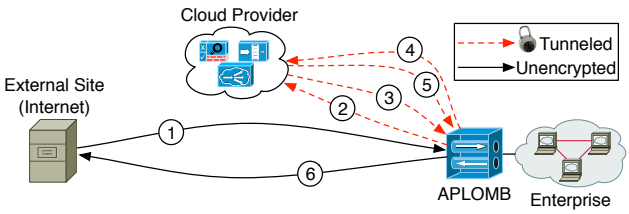
Further, certain 'extremely local' middleboxes such as proxies and WAN optimizers rely on being local to obtain significant reductions in latency and bandwidth costs. Caching proxies effectively terminate communication from an enterprise host a to an external host b thus reducing communication latency from that of path a-m-b to that of a-m. Likewise, WAN optimizers include a protocol acceleration component that achieves significant latency savings (although using very different mechanisms from a proxy).Thus, the latency optimizations we develop also must serve to minimize the latency increase due to taking extremely local middleboxes out of the enterprise.

APLOMB +: 'Extremely local' middleboxes not only reduce latency, but also reduce bandwidth consumption. Caching proxies, by serving content from a local store, avoid fetching data from the wide area; WAN Optimizers include a redundancy elimination component. To retain the savings in bandwidth consumption, we propose what we term APLOMB + appliances that extend APLOMB to provide comparable bandwidth reduction to extremely local appliances.

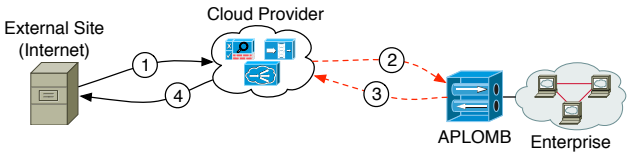
We explore solutions for the above design components in §3.1 (redirection), §3.2 (low latency) and §3.3 (APLOMB+). Recall from §1 that our design goals are to ensure: (i) functional equivalence, (ii) low performance overhead, and (iii) low enterprise-side complexity. We analyze our design options through the lens of these goals and recap the solution we arrive at in §3.4.

3.1 Redirection

We consider three natural approaches to redirection and discuss their latency vs. complexity tradeoffs.



(a) A "bounce" architecture requires excess RTTs.



(b) A direct architecture reduces latency.

Figure 5: Comparing two redirection architectures.

3.1.1 *Bounce Redirection*

In the simplest case, the APLOMB gateway at the enterprise tunnels both ingress and egress traffic to the cloud, as shown in Figure 5(a). Incoming traffic is bounced to the cloud PoP (1), processed by middleboxes, then sent back to the enterprise (2,3) and delivered to the appropriate hosts. Outgoing traffic is similarly redirected (4-6).

This scheme has two advantages. First, the APLOMB gateway is the only device that needs to be cloud-aware; no modification is required to existing enterprise network or application infrastructure. Second, the design requires minimal gateway functionality and configuration—a few static rules to redirect traffic to the PoP. The obvious drawback of this architecture is the increase in end-to-end latency due to an extra round trip to the cloud PoP for each packet.[4]

3.1.2 *IP-based Redirection*

To avoid the extra round-trips in bounce redirection, we might instead route traffic directly to/from the cloud as in Figure 5(b). One approach is to redirect traffic at the IP level: for example, the cloud provider could announce IP prefix P on the enterprise's behalf. Hosts communicating with the enterprise direct their traffic to P and thus their enterprise-bound traffic is received by the provider. The cloud provider, after processing the traffic, then tunnels the traffic to the enterprise gateways, who announce an additional prefix P'. [5]

In practice, enterprises would like to leverage the multi-PoP footprint of a provider for improved latency, load distribution and fault tolerance. For this, the cloud provider might advertise P from multiple PoPs so that client traffic is effectively 'anycasted' to the closest PoP. Unfortunately, IP-based redirection breaks down in a multi-PoP scenario

[4]We could eliminate a hop for outgoing traffic by routing return traffic directly from the cloud to the external target. However, this would require the cloud provider to spoof the enterprise's IP addresses, and such messages may be filtered by intermediate ISPs.
[5]The prefix P would in fact have to be owned by the cloud provider. If the cloud provider simply advertises a prefix assigned to the enterprise, then ISPs might filter the BGP announcements as they would fail the origin authorization checks.

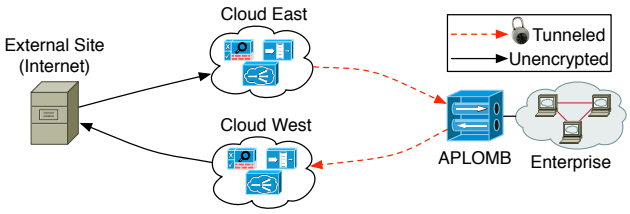


Figure 6: A pure-IP solution cannot ensure that inbound and outbound traffic traverse the same PoP, breaking bidirectional middlebox services.

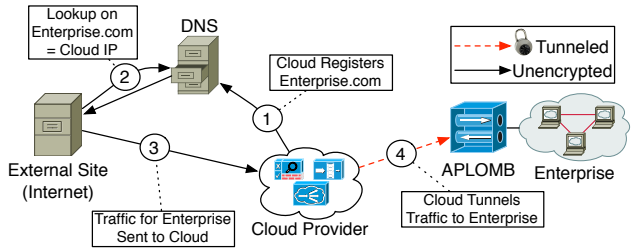


Figure 7: DNS redirection step by step.

since we cannot ensure that traffic from a client a to enterprise b will be routed to the same cloud PoP as that from b to a, thus breaking stateful middleboxes. This is shown in Figure 6 where the Cloud-West PoP is closest (in terms of BGP hops) to the enterprise while Cloud-East is closest to the external site. Likewise, if the underlying BGP paths change during a session then different PoPs might be traversed, once again disrupting stateful processing. Finally, because traffic is redirected at the network layer based on BGP path selection criteria (*e.g.*, AS hops), the enterprise or the cloud provider has little control over which PoP is selected and cannot (for example) pick PoPs to optimize end-to-end latency. Because of these limitations, we reject IP-based redirection as an option.

3.1.3 *DNS-based Redirection*

DNS-based redirection avoids the problems of IP-based redirection. Here the cloud provider runs the DNS resolution on the enterprise's behalf [3]. We explain this using the example in Figure 7. After an enterprise client provides its cloud provider with a manifest of their externally accessible services, the provider registers DNS names on behalf of the client's external services (step 1); *e.g.*, the provider registers '`MyEnterprise.com`'. When a user performs a DNS lookup on `MyEnterprise.com` (step 2), the DNS record directs it to the cloud PoP. The user then directs his traffic to the cloud PoP (step 3), where the traffic undergoes NAT to translate from the public IP address mapped to the cloud PoP to a private IP address internal to the enterprise client's network. The traffic is then processed by any relevant middleboxes and tunneled (step 4) to the enterprise.

This scheme addresses the bidirectionality concerns even in a multi-PoP setting as the intermediate PoP remains the same even if the network-level routing changes. Outbound traffic from the enterprise is relatively easy to control; the gateway device looks up a redirection map to find the PoP to which it must send return traffic. This ensures the symmetric traversal of middleboxes. Finally, Internet traffic initiated by enterprise hosts undergo NAT at the cloud provider. Thus, return traffic is forced to traverse the same PoP based on the public IP the provider assigned this connection.[6]

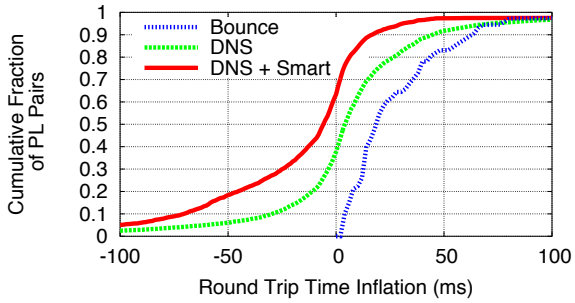


Figure 8: Round Trip Time (RTT) inflation when redirecting traffic between US PlanetLab nodes through Amazon PoPs.

3.1.4 *Redirection Tradeoffs*

To compare the latency inflation from bounce redirection *vs.* DNS-based redirection, we use measurements from over 300 PlanetLab nodes and twenty Amazon CloudFront locations. We consider an enterprise "site" located at one of fifty US-based PlanetLab sites while the other PlanetLab nodes emulate "clients". For each site e, we pick the closest Amazon CloudFront PoP $P_e^* = \arg\min_P Latency(P, e)$ and measure the impact of tunneling traffic to/from this PoP.

Figure 8 shows that the simplest bounce redirection can increase the end-to-end RTT by more than 50ms for 20% of inter-PlanetLab paths. The basic DNS-based redirection reduces the 80^{th} percentile of latency inflation $2\times$ compared to bounce redirection. In fact, for more than 30% of the pairwise measurements, the latency is actually lower than the direct IP path. This is because of well-known triangle inequality violations in inter-domain routing and the fact that cloud providers are very well connected to tier-1/2 ISPs [35]. Hence because the additional enterprise-side complexity required for DNS-based redirection is minimal and yet it achieves significantly lower latencies than Bounce redirection, we choose the DNS-based design.

3.2 Low Latency Operation

We now consider additional latency-sensitive PoP selection algorithms and analyze the scale of deployment a cloud provider requires to achieve low latency operation.

3.2.1 *Smarter Redirection*

So far, we considered a simple PoP selection algorithm where an enterprise site e picks its closest PoP. Figure 8 shows that with this simple redirection, 10% of end-to-end scenarios still suffer more than 50ms inflation. To reduce

[6]Many enterprises already use NATs to external services for other reasons (*e.g.*, flexibility and security); we introduce no new constraints.

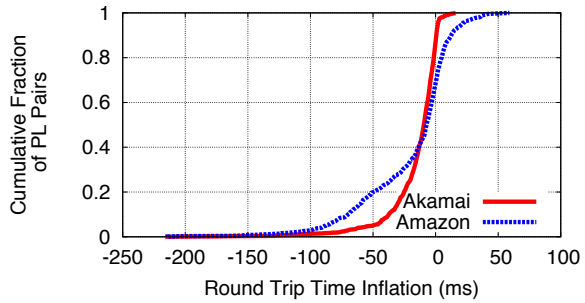


Figure 9: PlanetLab-to-PlanetLab RTTs with APLOMB redirection through Amazon and Akamai.

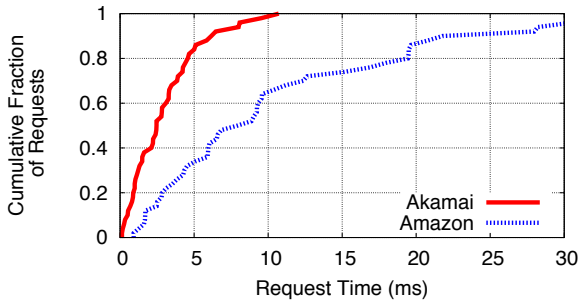


Figure 10: Direct RTTs from PlanetLab to nearest Akamai or Amazon redirection node.

this latency further, we will try to utilize multiple PoPs from the cloud provider's footprint to optimize the *end-to-end* latency as opposed to just the enterprise-to-cloud latency. That is, instead of using a single fixed PoP P_e^* for each enterprise site e, we choose the optimal PoP for each c, e combination. Formally, for each client c and enterprise site e, we identify:

$$P_{c,e}^* : arg \min_P Latency(P, c) + Latency(P, e)$$

We quantify the inflation using smart redirection and the same experimental setup as before, with Amazon CloudFront sites as potential PoPs and PlanetLab nodes as enterprise sites. Figure 8 shows that with this "Smart Redirection", more than 70% of the cases have zero or negative inflation and 90% of all traffic has less than 10ms inflation.

Smart redirection requires that the APLOMB appliance direct traffic to different PoPs based on the client's IP and maintain persistent tunnels to multiple PoPs instead of just one tunnel to its closest PoP. This requirement is modest: mappings for PoP selection can be computed at the cloud provider and pushed to APLOMB appliances, and today's commodity gateways can already support hundreds of persistent tunneled connections.

Finally, we note that if communication includes extremely local appliances such as proxies and WAN optimizers, then the bulk of communication is between the enterprise and the middlebox and hence the optimal strategy (which we follow) for such cases is still to simply pick the closest PoP.

3.2.2 Provider Footprint

We now analyze how the middlebox provider's choice of geographic footprint may impact latency. Today's clouds have a few tens of global PoPs and expand as new demand arises [5]. For greater coverage, we could envision an extreme point with a middlebox provider with a footprint comparable to CDNs such as Akamai with thousands of vantage points [47]. While it is clear that a larger footprint provides lower latency, what is not obvious is how large a footprint is required in the context of outsourcing middleboxes.

To understand the implications of the provider's footprint, we extend our measurements to consider a cloud provider with an Akamai-like footprint using IP addresses of over 20,000 Akamai hosts [30]. First, we repeat the the end-to-end latency analysis for paths between US PlanetLab nodes and see that a larger, edge-concentrated Akamai footprint reduces tail latency, but the overall changes are marginal compared to a smaller but well connected Amazon-like footprint. End-to-end latency is the metric of interest when outsourcing most middleboxes – all except for 'extremely local' appliances. Because roughly 70% of inter-PlanetLab node paths actually experience *improved* latency, these results suggest that a middlebox provider can service most customers with most types of middleboxes (*e.g.*, NIDS, firewalls) with an Amazon-like footprint of a few tens of PoPs.

To evaluate whether we can outsource even extremely local middleboxes without a high latency penalty (we discuss bandwidth penalties in §3.3), we look at the RTT between each Planetlab node and its closest Akamai node in Figure 10. In this case, we see a more dramatic impact of Akamai's footprint as it provides sub-millisecond latencies to 20% of sites, and less than 5 ms latencies to almost 90% of sites. An Amazon-like footprint provides only 30% of sites with an RTT <5 ms. Hence our results suggest that an Amazon-like footprint can serve latency acceleration benefits in only a limited portion of the US; to serve a nation-wide set of sites, an Akamai-like footprint is necessary.

3.3 APLOMB+ Gateways

As mentioned earlier, extremely local appliances optimize both latency and bandwidth consumption. Our results above suggest that, with an appropriate provider footprint, these appliances can be outsourced and still offer significant latency savings. We now consider the question of the bandwidth savings they enable. Unfortunately, this is a harder problem since bandwidth optimizations must fundamentally be implemented before the enterprise access link in order to be useful. We thus see three options, described below.

The first is to simply not outsource these appliances. From the enterprises we surveyed and Figure 1, we see that WAN optimizers and proxies are currently only deployed in large enterprises and that APLOMB is of significant value even if it doesn't cover proxies and WAN optimizers. Nevertheless, we'd like to do better and hence ask whether a full-fledged middlebox is really needed or whether we could achieve much of their benefit with a more minimal design.

Thus the second option we consider is to embed some *general-purpose* traffic compression capabilities into the APLOMB

appliance—we term such an augmented appliance an APLOMB+. In §5.3, we evaluate APLOMB+ against traditional WAN optimizers using measurements from a large enterprise and show that protocol-agnostic compression [25] can provide similar bandwidth savings (Figure 18). While our measurements suggest that in the specific case of WAN optimization a minimalist APLOMB+ suffices, we do not claim that such a minimal capability exists for every conceivable middlebox (*e.g.*, consider an appliance that encodes outgoing traffic for loss protection), nor that APLOMB+ can fully replicate the behavior of dedicated appliances.

Our third option considers more general support for extremely local appliances at the APLOMB gateway. For this, we envision a more "active" appliance architecture that can run specialized software modules (*e.g.*, a FEC encoder). A minimal set of such modules can be dynamically installed either by the cloud provider or the enterprise administrator. Although more general, this option increases both device and configuration complexity for the enterprise. For this reason, and because APLOMB+ suffices to outsource the extremely local appliances we find in today's networks, we choose to implement APLOMB+ in our design.

Type of Middlebox	Enterprise Device	Cloud Footprint
IP Firewalls	Basic APLOMB	Multi-PoP
Application Firewalls	Basic APLOMB	Multi-PoP
VPN Gateways	Basic APLOMB	Multi-PoP
Load Balancers	Basic APLOMB	Multi-PoP
IDS/IPS	Basic APLOMB	Multi-PoP
WAN optimizers	APLOMB+	CDN
Proxies	APLOMB+	CDN

Table 2: Complexity of design and cloud footprint required to outsource different types of middleboxes.

3.4 Summary

We briefly recap our design and its performance and complexity tradeoffs. At the enterprise end, the functionality we require is embedded in an APLOMB appliance. The basic APLOMB tunnels traffic to multiple cloud PoPs and stores a redirection map based on which it forwards traffic to the cloud. The cloud provider uses DNS redirection to redirect traffic from the enterprise's external contacts to a cloud PoP before forwarding it to the enterprise. APLOMB+ augments this basic functionality with general compression for bandwidth savings.

In addition to middlebox processing, a cloud-based middlebox provider must support DNS translation for its customers, NAT, and tunneling. The key design choice to a provider is the scale of its deployment footprint. We saw that an Amazon-like footprint often *decreases* latency relative to the direct IP path. However, for performance optimization devices, we saw that a larger Akamai-like footprint is necessary to provide extremely local services with nation-wide availability.

Table 2 identifies the design option (and hence its associated complexity) that is needed to retain the functional equivalence of the different middleboxes observed in our survey, *e.g.*, outsourcing an IP firewall requires only a basic APLOMB at the enterprise and an Amazon-scale footprint.[7]

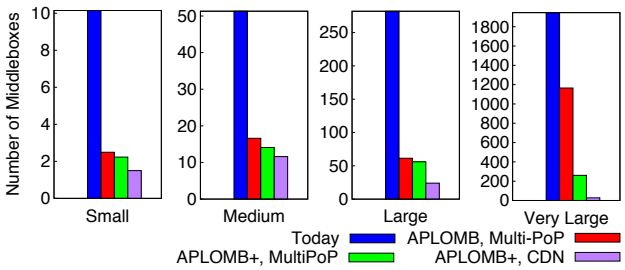


Figure 11: Average number of middleboxes remaining in enterprise under different outsourcing options.

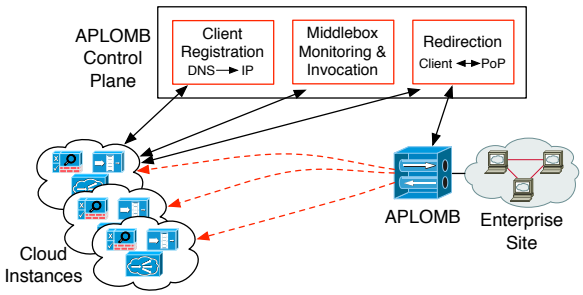


Figure 12: Architectural components of APLOMB.

Based on this analysis, Figure 11 shows the number of middleboxes that remain in an average small, medium, and large enterprise under different outsourcing deployment options. This suggests that small and medium enterprises can achieve almost all outsourcing benefits with a basic APLOMB architecture using today's cloud providers (we discuss the remaining middleboxes, 'internal firewalls', in §5.3). The same basic architecture can outsource close to 50% of the appliances in very large enterprise networks; using APLOMB+ increases the percentage of outsourced appliances to close to 90%.

4. APLOMB: DETAILED DESIGN

The APLOMB architecture realizes the key design choices we derived in the previous section: redirection based on DNS and NAT with fine-grained PoP selection for best performance. At a high level, we organize APLOMB into three components, illustrated in Figure 12. Enterprise networks deploy *APLOMB gateways* at their egress. These devices use configurable forwarding policies to tunnel traffic to multiple *cloud PoPs* hosting middlebox services for the client. Finally, the core logic and state reside in the *APLOMB Control Plane* which tracks clients and middleboxes, selects routes, launches new middlebox instances according to demand, and manages and configures middlebox routing policies in the cloud. We now discuss each of these components.

[7]We note that even load balancers can be outsourced since APLOMB retains stateful semantics. One subtle issue is whether load balancers really need to be physically close to backend servers; *e.g.*, for identifying load imbalances at the sub-millisecond granularity. Our conversations with administrators suggest that this is not a typical requirement.

4.1 APLOMB Gateways and the Enterprise

Redirecting traffic from the enterprise client to the cloud middlebox provider is simple: an APLOMB gateway is co-located with the enterprise's gateway router, and enterprise administrators supply the cloud provider with a manifest of their address allocations. APLOMB changes neither routing nor switching, and end hosts require no new configuration.

Address Manifests. Initial configuration for outsourcing requires a limited registration step in which administrators provide the cloud provider with an *address manifest*. Address manifests list the enterprise network's address blocks in its *private* address space and associates each address or prefix with one of three types of address records:

Protected services: Most private IP addresses are registered as protected services. These address records contain an IP address or prefix and the public IP address of the APLOMB device at the gateway to the registered address(es). This registration allows inter-site enterprise traffic to traverse the cloud infrastructure (*e.g.* a host at site A with address 10.2.3.4 can communicate with a host at site B with address 10.4.5.6, and the cloud provider knows that the internal address 10.4.5.6 maps to the APLOMB gateway at site B). The cloud provider allocates no permanant public IP address for hosts with 'protected services' addresses; Internet-destined connections instead undergo traditional NAPT.

DNS-based services: For hosts which accept incoming traffic, such as web servers, a publicly routeable address must direct incoming traffic to the appropriate cloud PoP. For these IP addresses, the administrator requests DNS service in the address manifest, listing the private IP address of the service, the relevant APLOMB gateway, and a DNS name. The cloud provider then manages the DNS records for this address on the enterprise client's behalf. When a DNS request for this service arrives (see Figure 7), the cloud provider (dynamically) assigns a public IP from its own pool of IP addresses and directs this request to the appropriate cloud PoP and subsequent APLOMB gateway.

Legacy IP services: While DNS-based services are the common case, enterprise may require legacy services that require fixed IP addresses. For these services, the enterprise registers the internal IP address and corresponding APLOMB gateway, and the cloud provider allocates a static public IP address at a single PoP for the IP service. For this type of service, we fall back to the single-PoP Cloud-IP solution rather than DNS redirection discussed in §3.

The APLOMB Gateway. The APLOMB gateway is logically co-located with the enterprise's gateway router, and the functionality required of the APLOMB gateway is simple enough that it can be bundled with the egress router itself.

Each APLOMB gateway supports two functions: maintaining persistent tunnels to multiple cloud PoPs and keeping a routing table to direct outgoing traffic to the appropriate cloud PoP. The gateway registers itself with the cloud controller (§4.3), which supplies it with a list of cloud tunnel endpoints in each PoP and policy-based route (PBR [8]) rules (5-tuple $\rightarrow$ cloud PoP Identifier) for redirection. The gateway router must not allow any ingress IP traffic into the network that is not destined for an APLOMB gateway.

Load balancing traffic across multiple APLOMB gateways is simple. To load balance ingress traffic, the enterprise's private address space can be split to direct traffic to, *e.g.* 10.1.0.0/17 to one gateway, and 10.1.128.0/17 to another. To handle gateway failures, we can equip the APLOMB hardware with fail-open NICs and configure the NICs to direct the packets to a APLOMB replica under failure. Since each APLOMB box keeps almost no per-flow state, the replica receiving traffic from the failed device can start forwarding the new traffic without interruption to existing flows.

Maintaining a tunnel to a cloud PoP can be as simple as using IP-in-IP encapsulation; the tunnel can also be encrypted (for security) and compressed (to reduce bandwidth costs). APLOMB uses protocol- and application-agnostic redundancy elimination [25] to reduce bandwidth consumption. Hardware requirements are modest: enterprises use commodity x86 hardware for similar functionality today [15].

4.2 Cloud PoPs

The APLOMB architecture requires that cloud PoPs (1) map publicly addressable IP addresses to the appropriate enterprise customer and internal private address, (2) apply middlebox processing services to the customers' traffic according to their *policies* (§4.3), and (3) tunnel traffic to and from the appropriate APLOMB gateways at enterprise sites. Thus, the core components at the cloud PoP are:

- *Tunnel Endpoints* to encapsulate/decapsulate traffic from the enterprise (and to encrypt/decrypt and compress/decompress if enabled)
- *Middlebox Instances* to process the customers' traffic
- *NAT Devices* to translate between publicly visible IP addresses and the clients' internal addresses. NAT devices manage both statically configured IP to IP mappings for DNS and Legacy IP services, and generate IP and port mappings on the fly for Protected Services (§4.1).
- A *Routing Layer* to route between the above three.

The cloud controller (§4.3) configures each component. In Section 4.4 we discuss our implementation, which implements all of these features on a public cloud. The APLOMB architecture, however, is agnostic to how the cloud provider hosts and manages middleboxes, so long as they can implement the above four components.

Solutions already exist to provide the required features, *e.g.* [37, 29, 10]. These solutions make possible a broad range of cloud deployment scenarios. Certainly, deploying physical hardware appliances [17, 13] requires physical access to a datacenter. Given the prevalence of production-grade software middleboxes [43, 48, 16], forwarding rates of high performance servers [32, 44], and cloud offerings for custom virtual network topologies [4], it is also feasible that the 'middlebox service provider' is a customer of a public cloud. The latter has the benefit of allowing the middlebox

provider to leverage the footprint of a large cloud provider with a geographically diverse datacenter deployment. Our implementation follows the second model but is only one of many ways to build cloud middlebox services.

4.3 Control Plane

A driving design principle for APLOMB is to keep the new components introduced by our architecture that are on the critical path, the APLOMB gateway device and the cloud terminal endpoint, as simple and as stateless as possible. This not only reduces the enterprise's administrative overhead but also enables seamless transition in the presence of hardware and network failures. Contrastingly, the Control Plane must manage all of the state representing active APLOMB gateways, cloud PoPs and middlebox instances, and tunnel endpoints. It is then responsible for the complexity of determining optimal redirection routes between communicating parties, managing and pushing middlebox policy configurations, and dynamically scaling cloud middlebox capacity to meet demands.

In practice, the control plane is realized in a *cloud controller*, which manages every APLOMB gateway, middlebox, tunnelling end point, and cloud routing policy in the APLOMB architecture.[8] On startup, each entity (APLOMB device, middlebox, *etc.*) registers itself with the controller. The controller sends periodic 'heartbeat' health checks to each device to verify its continued activity. In addition, the controller gathers RTTs from each PoP to every prefix on the Internet (for PoP selection) and utilization statistics from each middlebox (for adaptive scaling). We now discuss the redirection optimization, policy management, and middlebox scaling performed by the cloud controller.

Redirection Optimization. Using measurement data from the cloud PoPs, the cloud controller pushes the current best (as discussed in §3.2) redirection routes to the APLOMB gateways at the enterprise and mappings in the DNS. To deal with transient routing failures or performance instability, the cloud controller periodically updates routes based on the newest measurements from each cloud PoP.

Policy Configuration. The cloud controller is also responsible for managing and pushing routing *policies*, thus, the cloud provider implements a rich policy configuration interface [36]. This policy interface exports the available types of middlebox processing to enterprise administrators and also a programmatic interface to specify the types of middlebox processing required. The enterprise administrators can request a *policy chain* of middlebox processing for each class of traffic, defining a class of traffic using the traditional 5-tuple categorization of flows (i.e., source and destination IPs and port values along with the protocol type). For example, an enterprise could require all egress traffic to go through a firewall → exfiltration engine → proxy. Or, that

[8]While the cloud controller may be in reality a replicated set of federated controllers, for simplicity this discussion simply refers to a single controller.

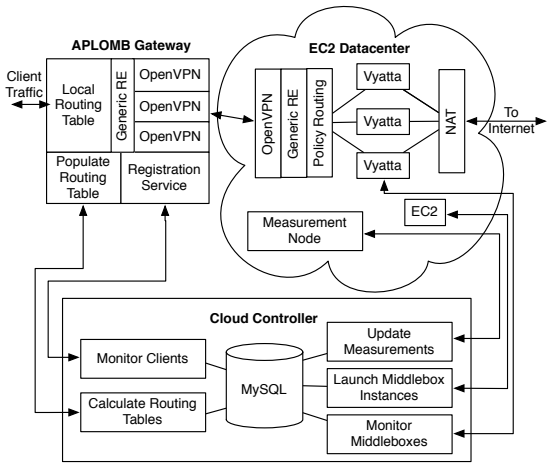


Figure 13: Software architecture of APLOMB.

all ingress traffic traverse a firewall → IDS, and all traffic to internal web services further go through an application-level firewall. If appropriate, the provider may also export certain device-specific configuration parameters that the enterprise administrator can tune.

In the common case, we expect the policy churn rate to be low and to roughly mirror existing middlebox deployments. However, the APLOMB configuration interface also allows administrators to dynamically invoke new or additional capabilities, *e.g.* invoking a packet scrubbing service when they suspect they are under attack [7].

Middlebox Scaling. Because middlebox processing is opaque to the client, APLOMB providers have a great deal of flexibility in how they actually implement the desired middlebox processing. In particular, as utilization increases on a particular middlebox, the APLOMB provider simply increases the number of instances of that middlebox being utilized for a client's traffic and updates policy routing and load balancers without any observable disruption to the client.

Using data from heartbeat health checks on all middleboxes, the cloud controller detects changes in utilization. When utilization is high, the cloud controller launches new middleboxes and updates the policy routing framework; when utilization is low, the cloud controller deactivates excess instances. While scaling is simpler if all middlebox processing is performed in software on standard virtual machines, providers using hardware middleboxes could achieve the same result using policy routing alone. Techniques for dynamic scaling under load are well-known for cloud computing applications like web servers [14]; as such we do not go in to detail here.

4.4 Implementation

To explore an APLOMB architecture in practice, we built a prototype system for cloud middlebox processing; we deployed middlebox processing services on EC2 and APLOMB endpoints in our lab and at the authors' homes. We consciously choose to use off-the-shelf components that run on existing cloud providers and end host systems; this makes our system easy to deploy and use, demonstrating that bar-

riers to adoption are minimal. Our APLOMB endpoint software can be deployed on a stand-alone software router or as a tunneling layer on an end host; installing and running the end host software is as simple as connecting to a VPN.

Figure 13 is a software architecture diagram of our implementation. We implement a cloud controller on a server in our lab and use geographically distributed EC2 datacenters as cloud PoPs. Our cloud controller employs a MySQL database to store data on active middlebox nodes, RTTs to and from cloud PoPs, and registered clients. The cloud controller monitors APLOMB devices, calculates and pushes routing tables to the APLOMB devices, requests measurements from the cloud PoPs, monitors middlebox instances, and scales middlebox instances up or down as demand varies.

At the enterprise or the end host, the APLOMB maintains several concurrent VPN tunnels, one to a remote APLOMB each cloud PoP. On startup, the APLOMB software contacts the cloud controller and registers itself, fetches remote APLOMB endpoints for each cloud PoP, and requests a set of initial routes. A simple routing layer, populated by the cloud controller, directs traffic to the appropriate endpoint tunnel, and a redundancy elimination encoding module compresses all outgoing traffic. Running on a software router, ingress traffic comes from an attached hosts for whom the router serves as their default gateway. Running on a laptop or end host, static routes in the kernel direct application traffic to the appropriate egress VPN tunnel.

EC2 datacenters host tunnel endpoints, redundancy elimination decoders, middlebox routers, and NATs, each with an inter-device routing layer and controller registration and monitoring service. For tunneling, we use OpenVPN [12], a widely-deployed VPN solution with packages for all major operating systems. We use a Click [39] implementation of the redundancy elimination technique described by Anand et al [25]. For middlebox processing, we use Vyatta [20], a customizable software middlebox. Our default Vyatta installation performs firewalling, intrusion detection, caching, and application-layer web filtering. Policy configurations (§4.3) are translated in to Vyatta configurations such that each client can have a unique Vyatta configuration dependent on their needs. Finally, each cloud PoP also hosts one 'measurement node', which persistently issues ping measurements for RTT estimation to assist in PoP selection.

5. EVALUATION

We now evaluate APLOMB. First (§5.1), we present performance benchmarks for three common applications running over our implementation.We then (§5.2) demonstrate APLOMB's dynamic scaling capability and its resilience to failure.Having shown that APLOMB is practical, we return to our goal of outsourcing all middlebox functionality in an enterprise with a trace-driven evaluation of APLOMB outsourcing applied to data from a real middlebox deployment in a large enterprise (§5.3).

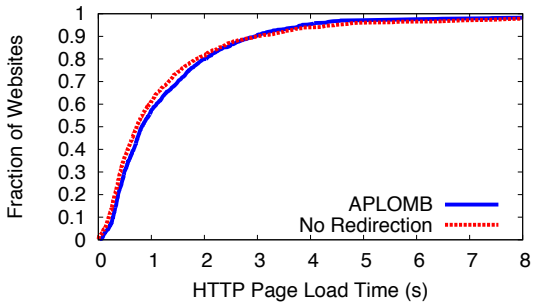


Figure 14: CDF of HTTP Page Load times for Alexa top 1,000 sites with and without APLOMB.

5.1 Application Performance

We first demonstrate that APLOMB's architecture is practical for enterprise use with performance benchmarks for common applications using our APLOMB implementation.

HTTP Page Loads: In Figure 14, we plot page load times (fetching the front page and all embedded content) from a university network for the Alexa [1] top 1,000 most popular web pages with and without APLOMB processing. We performed this experiment with a vacant cache. For pages at the 50^{th} percentile, page loads without APLOMB took 0.72 seconds, while page loads with took 0.82 seconds. For pages at the 95^{th} percentile, using APLOMB results in shorter page load times: 3.85 seconds versus 4.53 seconds.

BitTorrent: While we don't expect BitTorrent to be a major component of enterprise traffic, we chose to experiment with BitTorrent because it allowed us to observe a bulk transfer over a long period of time, to observe many connections over our infrastructure simultaneously, and to establish connections to non-commercial endpoints. We downloaded a 698MB public domain film over BitTorrent with and without APLOMB from both a university network and from a residential network, five times repeatedly. The average residential download took 294 seconds without APLOMB, with APLOMB the download speed increased 2.8% to 302 seconds. The average university download took 156 seconds without APLOMB, with APLOMB the average download took 165 seconds, a 5.5% increase.

Voice over IP: Voice over IP (VoIP) is a common enterprise application, but unlike the previously explored applications, VoIP performance depends not only on low latency and high bandwidth, but on low *jitter*, or variance in latency. APLOMB easily accommodates this third demand: we ran VoIP calls over APLOMB and for each call logged the jitter estimator, a running estimate of packet interarrival variance developed for RTP. Industry experts cite 30ms of one-way jitter as a target for maximum acceptable jitter [9]. In the first call, to a residential network, median inbound/outbound jitter with APLOMB was 2.49 ms/2.46 ms and without was 2.3 ms/1.03 ms. In the second, to a public WiFi hotspot, the median inbound/outbound jitter with APLOMB was 13.21 ms/14.49 ms and without was 4.41 ms/4.04 ms.

5.2 Scaling and Failover

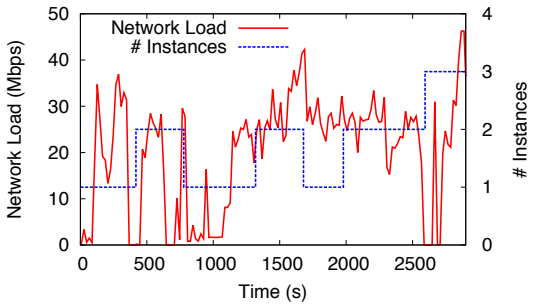


Figure 15: Network load (Y_1) and number of software middlebox instances (Y_2) under load. Experiment used low-capacity instances to highlight scaling dynamics.

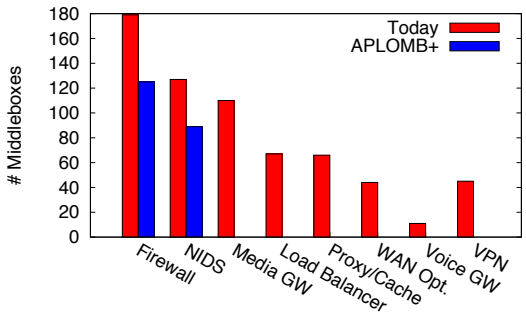


Figure 16: Number of middleboxes in the enterprise with and without APLOMB+. The enterprise has an atypical number of 'internal' firewalls and NIDS.

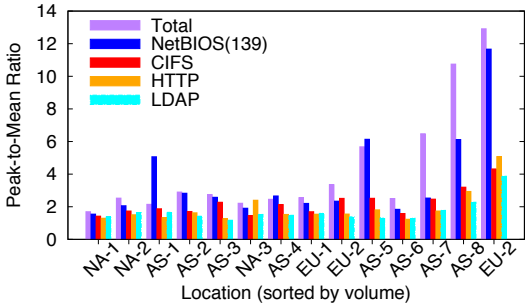


Figure 17: Ratio of peak traffic volume to average traffic volume, divided by protocol.

To evaluate APLOMB dynamic scaling, we measured traffic from a single client to the APLOMB cloud. Figure 15 shows capacity adapting to increased network load over a 10-minute period. The client workload involved simultaneously streaming a video, repeatedly requesting large files over HTTP, and downloading several large files via BitTorrent. The resulting network load varied significantly over the course of the experiment, providing an opportunity for capacity scaling. The controller tracks CPU utilization of each middlebox instance and adds additional capacity when existing instances exceed a utilization threshold for one minute.

While middlebox capacity lags changes in demand, this is primarily an artifact of the low sampling resolution of the monitoring infrastructure provided by our cloud provider. Once a new middlebox instance has been allocated and initialized, actual switchover time to begin routing traffic through it is less than 100ms. To handle failed middlebox instances, the cloud controller checks for reachability between itself and individual middlebox instances every second; when an instance becomes unreachable, APLOMB ceases routing traffic through it within 100ms. Using the same mechanism, the enterprise APLOMB can cope with failure of a remote APLOMB, re-routing traffic to another remote APLOMB in the same or even different cloud PoP, providing fault-tolerance against loss of an entire datacenter.

5.3 Case Study

We set out with the goal of outsourcing as many middleboxes as possible and reducing enterprise costs, all the while without increasing bandwidth utilization or latency. To evaluate our success, we return to the very large enterprise data and perform a trace-driven analysis to determine:

- How many middleboxes can the enterprise outsource?
- What are the gains from elastic scaling?
- What latency penalty will inter-site traffic suffer?
- How much does the enterprise's bandwidth costs increase?

Middleboxes Outsourced: Figure 16 shows the aggregate number of middleboxes that the large enterprise can outsourced under APLOMB+ and a CDN footprint; close to 60% of the middleboxes can be outsourced.

This high fraction of outsourceability comes despite an atypically high deployment of "internal" firewalls and NIDS at this enterprise. Internal firewalls protect a host or subnetwork not only from Internet-originated traffic, but from traffic originated within the enterprise; the most common reason we found for these deployments was PCI compliance for managing credit card data. While the average enterprise of this size deploys 27.7 unoutsourceable internal firewalls, this enterprise deploys over 100 internal firewalls. From discussions with the network's administrators, we learned these were installed in the past to protect internal servers against worms that preferentially scanned internal prefixes, *e.g.* CodeRed and Nimda. We also learned that the enterprise is in the process of moving much of their computing infrastructure to the cloud; when this occurs, the internal firewalls protecting them could move to the cloud as well.

Cost Reduction: To evaluate benefits from elastic scaling, in Figure 17 we focus on each site of the enterprise and show the ratio of peak-to-average volumes for total inter-site traffic. The peak represents a conservative estimate of the traffic volume the enterprise has provisioned at the site, while the average is the typical utilization; we see that most sites are provisioned over 2× their typical load, and some of the smaller sites as much as 12×! In addition, we show peak-to-average values for the top four protocols in use. The per-protocol numbers are indicative of elasticity savings per middlebox, as different protocols are likely to traverse different middleboxes.

Latency: We measured redirection latency for inter-site traffic between the top eleven sites of the enterprise through the APLOMB infrastructure by pinging hosts at each site

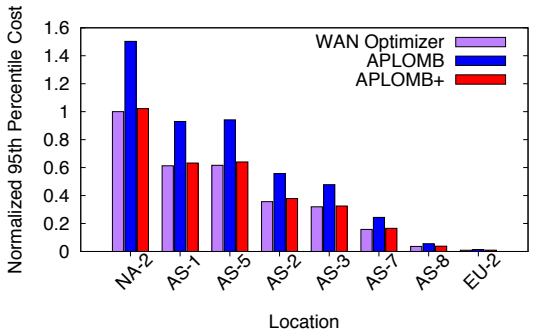


Figure 18: 95^{th} percentile bandwidth without APLOMB, with APLOMB, and with APLOMB+.

from within EC2. We found that for more than 60% of inter-site pairs, the latency with redirection is almost identical to the direct RTT. After digging into the cases where latency inflated, we found that most sites with inflated latency were in Asia, where EC2 does not have a wide footprint.

We also calculated a weighted inflation value, weighted by traffic volume and found that in expectation a typical redirected packet experiences only 1.13 ms of inflation. This results from the fact that the inter-site pairs with high traffic volume actually have negative inflation, by virtue of one or both endpoints being in the US or Europe, where EC2's footprint and connectivity is high.

Bandwidth Last, we evaluate bandwidth inflation. We ran an enterprise traffic trace through our APLOMB prototype with and without generic redundancy elimination. Without Generic RE, the bandwidth utilization increased by 6.2% due to encryption and encapsulation overhead. With Generic RE, the bandwidth utilization reduced by 28%, giving APLOMB+ a 32% improvement over APLOMB.

As we observed in §2, many larger enterprises already compress their inter-site traffic using WAN optimizers. To evaluate the impact of switching compression for inter-site traffic from a traditional WAN optimizer solution to APLOMB+, we compared our observed benefits to those provided by WAN optimizers at eight of the large enterprise sites. In Figure 18, we measure the bandwidth cost of a given site in terms of the 95^{th} percentile of the total traffic volume with a WAN Optimizer, with APLOMB, and with APLOMB+. With APLOMB, the worst case inflation is 52% in the median case and at most 58%; APLOMB+ improves this to a median case of 3.8% inflation and a worst case of 8.1%.

6. DISCUSSION

APLOMB complements the ongoing move by enterprises from locally-hosted and managed infrastructure to outsourced cloud infrastructure. A network administrator at one large enterprise we surveyed reported their company's management had issued a broad mandate to moving a significant portion of their IT infrastructure to the cloud. Federal government agencies are also rapidly moving their IT infrastructure to the cloud, in compliance with a mandate to adopt a "cloud first" policy for new services and to reduce the number of existing federal datacenters by 800 before 2015 [40]. Nevertheless, many enterprises plan to keep at least some local infrastructure, citing security and performance concerns for applications currently deployed locally [23]. In fact, multiple vendors now offer services for integrating public cloud services with enterprises' existing infrastructure [2, 19], facilitating so-called "hybrid clouds". Rather than managing middlebox infrastructure for in-enterprise assets and cloud assets separately, consolidating all middlebox processing into the cloud simplifies management of these new hybrid infrastructures.

A key factor in enterprise adoption of APLOMB is cost, and while APLOMB reduces the cost of middlebox infrastructure, it may increase bandwidth costs due to current cloud business models. Today, tunnelling traffic to a cloud provider necessitates paying for bandwidth twice – once for the enterprise network's access link, and again at the cloud provider. Nevertheless, this does not mean that APLOMB will double bandwidth costs for an enterprise. We observed earlier that redundancy elimination and compression can reduce bandwidth demands at the enterprise access link by roughly 30%. This optimization is not possible without redirection through a cloud PoP, and allows a lower capacity, less expensive access link to satisfy an enterprise's needs. In the long term, we envision that a dedicated cloud middlebox provider would use a different pricing structure than cloud providers today, who typically charge for the volume of data transfered rather than the 95^{th} percentile rate. Such a provider would also have access to cheaper wholesale bandwidth than an enterprise, further reducing bandwidth costs at the cloud PoP.

Adopting APLOMB brings with it the same security questions as have challenged cloud computing. These challenges have not stopped widespread adoption of cloud computing services, nor the willingness of security certification standards to certify cloud services (for example, services on Amazon EC2 can achieve PCI-1 compliance, the highest level of certification for storing credit card data). However, these challenges remain concerns for APLOMB and cloud computing in general. Just as cloud storage services have raised questions about providing a cloud provider unencrypted access to data, cloud middlebox services give the cloud provider unencrypted access to traffic flows. Although VMs and other isolation techniques aim to protect customers of a cloud service from other, malicious, customers, some have demonstrated in the cloud computing context information leakage, *e.g.* through side-channels [42]. APLOMB encrypts tunneled traffic to and from the enterprise to protect against man-in-the-middle attacks, and allocates each client it's own set of VMs for middlebox processing, but ultimately it will not appeal to companies whose security policies restrict them from cloud computing in general.

7. RELATED WORK

Our work contributes to and draws inspiration from a rich corpus of work in cloud computing, redirection services, and

network management.

Cloud Computing: The motivation for APLOMB parallels arguments made by Armbrust et al. [27] in favor of traditional cloud computing. APLOMB also adapts techniques from traditional cloud solutions, *e.g.* utilization monitoring and dynamic scaling [14], and DNS-based redirection to datacenters with optimal performance for the customer [46].

Middlebox Management: Others have tackled middlebox management challenges within the enterprise [36, 37, 28, 31, 44]. Their solutions offer insights we can apply for managing middleboxes within the cloud – *e.g.*, the policy-routing switch of Joseph et al. [37], the management plane of Ballani et al. [28], and the consolidated appliance of Sekar et al. [44]. None of these proposals consider moving middlebox management out of the enterprise entirely, as we do. Like us, ETTM [31] proposes removing middleboxes from the enterprise network but, where we advocate moving them to the cloud, ETTM proposes the opposite: pushing middlebox processing to enterprise end hosts. As such, ETTM still retains the problem of middlebox management in the enterprise. Sekar et al [44] report on the middlebox deployment of a single large enterprise; our survey is broader in scope (covering a range of management and failure concerns) and covers 57 networks of various scales.

Redirection Services: Traffic redirection infrastructures have been explored in prior work [26, 45, 49] but in the context of improving Internet or overlay routing architectures as opposed to APLOMB's goal of enabling middlebox processing in the cloud. RON showed how routing via an intermediary might improve latency; we report similar findings using cloud PoPs as intermediaries. Walfish et al. [49] propose a clean-slate architecture, DOA, by which end hosts explicitly address middleboxes.

Cloud Networking: Using virtual middlebox appliances [20] reduces the physical hardware cost of middlebox ownership, but has limited scalability and does little to improve configuration complexity. Aryaka [6] and ZScalar [22] are two startups that focus on protocol accelleration and intrusion detection in the cloud; to some extent our work can be viewed as an extreme extrapolation of their offers and we offer a comprehensive exploration and evaluation of such a trend. CloudNaaS [29] and startup Embrane [10] aim at providing complete middlebox solutions for enterprise services that are *already* in the cloud.

8. CONCLUSION

Outsourcing middlebox processing to the cloud solves the major problems caused by today's enterprise middlebox infrastructure: cost, management complexity, capacity rigidity, and failures. Our survey of 57 enterprise network managers guides the design of APLOMB, a practical system for middlebox processing in the cloud. APLOMB succeeds in outsourcing the vast majority of middleboxes from a typical enterprise network without impacting performance, making scalable, affordable middlebox processing accessible to enterprise networks of every size.

References

[1] Alexa: The web information company. http://www.alexa.com/.
[2] Amazon Direct Connect. http://aws.amazon.com/directconnect.
[3] Amazon Route 53. http://aws.amazon.com/route53.
[4] Amazon Virtual Private Cloud. http://aws.amazon.com/vpc.
[5] Amazon Web Services launches Brazil datacenters for its cloud computing platform. http://tinyurl.com/amazonbrazil.
[6] Aryaka WAN Optimization. http://www.aryaka.com.
[7] Cisco DoS Prevention. http://tinyurl.com/cicscodos.
[8] Cisco policy based routing. http://tinyurl.com/ciscopbr.
[9] Cisco: Quality of service design overview. http://www.ciscopress.com/articles/article.asp?p=357102.
[10] Embrane. http://www.embrane.com/.
[11] Network monitoring tools. http://tinyurl.com/nmtools.
[12] OpenVPN. http://www.openvpn.com.
[13] Palo alto networks. http://www.paloaltonetworks.com/.
[14] Rightscale Cloud management. http://www.rightscale.com/.
[15] Riverbed: Application Acceleration. http://www.riverbed.com.
[16] Riverbed virtual steelhead. http://tinyurl.com/6o3vn9k.
[17] Symantec: Data Loss Protection. http://www.vontu.com.
[18] Tivoli Monitoring Software. http://tinyurl.com/7ycs5xv.
[19] VMWare vCloud. http://vcloud.vmware.com.
[20] Vyatta Software Middlebox. http://www.vyatta.com.
[21] World enterprise network and data security markets. http://bit.ly/gYW4Us.
[22] ZScaler Cloud Security. http://www.zscaler.com.
[23] Cloud computing - 31 companies describe their experiences. http://bit.ly/Asvwa2, 2011.
[24] M. Allman and V. Paxson. TCP congestion control. RFC 5681.
[25] A. Anand, A. Gupta, A. Akella, S. Seshan, and S. Shenker. Packet Caches on Routers: The Implications of Universal Redundant Traffic Elimination. In *Proc. of SIGCOMM*, 2008.
[26] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *SOSP*, 2001.
[27] M. Armbrust et al. A view of cloud computing. *Commun. ACM*, April 2010.
[28] H. Ballani and P. Francis. Conman: a step towards network manageability. In *SIGCOMM*, 2007.
[29] T. Benson, A. Akella, A. Shaikh, and S. Sahu. Cloudnaas: a cloud networking platform for enterprise applications. In *Proc. SOCC*, 2011.
[30] D. R. Choffnes and F. E. Bustamante. Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems. In *SIGCOMM*, 2008.
[31] C. Dixon, H. Uppal, V. Brajkovic, D. Brandon, T. Anderson, and A. Krishnamurthy. ETTM: a scalable fault tolerant network manager. In *NSDI*, 2011.
[32] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy. Routebricks: exploiting parallelism to scale software routers. In *SOSP*, 2009.
[33] N. Dukkipati and N. McKeown. Why flow-completion time is the right metric for congestion control. *CCR*, January 2006.
[34] S. Floyd. HighSpeed TCP for large congestion windows. RFC 3649.
[35] K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall. Improving the reliability of Internet paths with One-hop Source Routing. In *Proc. OSDI*, 2004.
[36] D. Joseph and I. Stoica. Modeling middleboxes. *Network, IEEE*, 22(5), 2008.
[37] D. A. Joseph, A. Tavakoli, and I. Stoica. A policy-aware switching layer for data centers. In *SIGCOMM*, 2008.
[38] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *SIGCOMM*, 2002.
[39] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM ToCS*, August 2000.
[40] V. Kundra. 25 Point Implementation Plan to Reform Federal Information Technology Management. Technical report, US CIO, 2010.
[41] N. McKeown et al. OpenFlow: enabling innovation in campus networks. *CCR*, March 2008.
[42] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *CCS*, 2009.
[43] M. Roesch. Snort - Lightweight Intrusion Detection for Networks. In *LISA*, 1999.
[44] V. Sekar, S. Ratnasamy, M. K. Reiter, N. Egi, and G. Shi. The middlebox manifesto: enabling innovation in middlebox deployment. In *HotNets*, 2011.
[45] I. Stoica et al. Internet indirection infrastructure. *ToN*, April 2004.
[46] A. Su, D. Choffnes, A. Kuzmanovic, and F. Bustamante. Drafting behind akamai (travelocity-based detouring). In *SIGCOMM*, 2006.
[47] V. Valancius, N. Laoutaris, L. Massouli'e, C. Diot, and P. Rodriguez. Greening the internet with nano data centers. In *Proc. CoNEXT*, 2009.
[48] Visolve. Transparent caching using Squid. http://www.visolve.com/squid/whitepapers/trans_caching.pdf, 2006.
[49] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes no longer considered harmful. In *OSDI*, 2004.