Design and Implementation of Practical Software Middleboxes

Summary

University Polithenica of Bucharest

Submitted for the degree of

Doctor of Philosophy

Vladimir Olteanu

Supervisor: Valentin Cristea

Keywords

- networking
- datacenter
- middlebox
- protocol

Summary

This thesis discusses the design of horizontally scalable software middleboxes.

Middleboxes are machines that process network packets, performing tasks as simple as basic filtering, or as complex as detecting leakage of sensitive information. They are a staple of today's Internet.

Traditionally, middleboxes were monolithic hardware appliances. Upgrading them either in terms of performance or functionality was typically costly, as it meant outright replacing the machine with a different model. Moreover, they had to cope with peak hour traffic; during off-peak hours the extra capacity was essentially wasted.

To address these issues, a new field of research emerged, called Network Function Virtualization. The key proposal is to move traffic processing from dedicated appliances to software running in a distributed manner on commodity hardware. Such middleboxes can scale out or in on the fly as the demand for capacity increases or decreases: extra capacity can be conjured up by adding more machines, and then removed by powering down some of them.

The most basic obstacle to scaling out middleboxes is that they may hold state, both per-flow and shared across multiple flows. Such state is key to the correct handling of the traffic; were it to become unavailable, even temporarily, all corresponding flows may experience disruptions. Therefore, scaling out not only entails redirecting a portion of the incoming traffic to another machine, but also migrating all relevant state to said machine.

There have been efforts to find a general framework that enables near-seamless scale-up and scale-down events, but they tend to incur a rather steep performance cost. The subtext of this thesis is that we argue against such generalized solutions; to that end, we design two separate middleboxes, and use tailor-made solutions to solve their scalability issues.

The first middlebox we discuss is a large scale NAT. It is a rather straightforward middlebox to implement, that holds both per-flow and per external IP. Our version features a novel state migration algorithm aimed at incurring a minimal performance penalty. Here we show that by simply risking reordering some packets while keeping all other guarantees intact, the performance cost of scale-out or scale-in events is minimal. Occasional packet reorderings are generally something that protocols at Layer 4 and up can cope with easily.

The second middlebox we discuss is Beamer, a stateless cloud-scale load-balancer, that also works with MPTCP. Stateful load-balancers remember associations between connections and back-end servers. This, in essence, attempts to duplicate information kept by the servers themselves in their connection tables. By not holding any per-flow state, Beamer muxes are simple to implement both in software and hardware, and avoid many of the pitfalls of scaling up and down. They can

easily handle SYN floods, something stateful designs struggle with.

Finally, we will discuss something that is a middlebox only in the loose sense of the word: we have designed version 6 of the decades-old SOCKS protocol, and built a proxy around it. Version 6 addresses many of the shortcomings of version 5: its RTT overhead in most cases is zero (or even negative!), it offers DNS from the proxy's vantage point, and it plays well with the new features introduced in TLS 1.3. Since the proxy is technically a service running on top of TCP, large SOCKSv6 deployments can be made using Beamer.

Contents