Exploiting Network Choice for Better Application QoE

Srikanth Sundaresan, Narseo Vallina-Rodriguez, Christian Kreibich
International Computer Science Institute, Berkeley
Lastline

1 Abstract

End-hosts today support a wide variety of networking capabilities, including protocol implementations, physical connectivity, and choice in services such as DNS providers. However, applications do not have the ability to make optimal use of these alternatives by dynamically adapting them to application requirements or changing network conditions: once developers have hardwired protocol policies into an application (*e.g.* by always favoring HTTP/2 over HTTP) or a particular configuration gets installed on an end-host, these decisions remain immutable, at best offering fallback capability to overcome outages. This state of affairs is sub-optimal because, intuitively, no single protocol or infrastructure can by itself deliver optimum performance under all conditions.

In this paper, we argue that the network stack needs to become more flexible in order to enable applications to dynamically leverage available protocol choice across the network stack to improve performance. To this end, we explore **PolyStack**, an end-host-centric architecture that extends the current protocol stack to intelligently adapt network configuration and protocol composition based on available choices in order to optimize performance. PolyStack complements current paradigms that focus on middleboxes and the cloud to optimize performance and provide better QoE. PolyStack constitutes one point in a much larger QoE-aware design space; in shifting focus back to the end-host and the application, they can exploit a maximum of possible avenues for optimizing performance.

2 A more flexible protocol stack

A significant amount of Internet research and innovation today focuses on networking paradigms to improve performance and bring more flexibility to networked systems. The rise of middleboxes and Network Functions Virtualization (NFV) architectures has provided network operators and designers with a rich and powerful suite of tools to manage and optimize applications and networks at relatively low operational and deployment costs. While one can implement much functionality efficiently within the network, such approaches ignore the potential benefits we can gain from intelligent composition of networking capabilities on the end-host. The edge of the network, where end-hosts and applications reside, is rich in choice. Any modern device supports a plethora of application- and transport-layer protocols, can configure multiple network resources such as DNS resolvers or HTTP proxies, and communicates via physical media such as WiFi and cellular uplinks. However, the current network stack can *not* act on this choice dynamically; once the application developer chooses a specific protocol preference, or a specific service configuration gets deployed on the end-host, its combination remains immutable (at least until these configurations get updated). A browser may always resort to HTTP/2 over HTTPS when the server supports it, and the local DNS resolver will stick to the configured resolvers despite better available alternatives only considered as fall-backs. This behavior is a direct consequence of protocol stack layering, whose central design tenet was to hide lower-layer complexity from the higher layers and focus on clean layer interfaces. Unfortunately, as widely reported by the research community, this strict layered design limits the ability of applications to adapt to evolving network conditions, particularly where a given protocol or network element may outperform an equivalent one. The result is suboptimal application QoE for the user.

We use DNS to illustrate our point. Suppose that DNS provider A provides lower-latency CDN replicas than DNS provider B for a subset of popular domains, whereas B provides better replicas for another subset. In such a situation we ideally should use the better resolver for the domain in question. We can extend this intuition to any other protocol and network element—for some endpoints IPv4 connectivity may prove better than IPv6, and at times resorting to cellular uplinks may outperform WiFi. It is unlikely that any of them, no matter how well designed, will outperform other protocols under *all* settings, but except for isolated optimizations (such

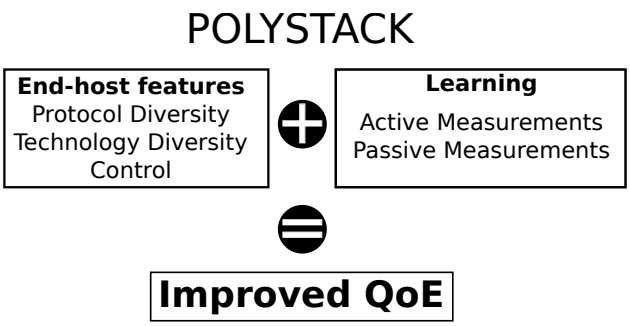

Figure 1: PolyStack leverages protocol and technology diversity on the end-hosts and combines it with learning capabilities to select the right protocol combination in order to optimize data transmissions.

as dynamic multi-homing or the "Happy Eyeballs" name lookup implementation recipe) this mindset precisely describes how protocol composition operates today.

3 Research Agenda

In this body of work, we argue for leveraging protocol and network resource diversity *holistically*, to provide more flexibility and adaptability in networked systems, particularly end-hosts. We are working on PolyStack, a framework for an intelligent and versatile protocol stack that combines an application's supported app-layer protocol portfolio and the state of the network in order to optimize performance and QoE by using available choice. We envision that PolyStack gathers information about the network using a combination of passive and active measurements and uses this information to intelligently adapt application traffic to optimize performance.

Designing PolyStack raises a number of challenges that we briefly list in the following.

- **How to learn?** Numerous questions arise when considering how to structure network and traffic measurements and learning. While every established connection automatically provides passive measurement opportunity, the volume of activity conducted "natively" by the client is unlikely to explore the choice points sufficiently, calling for complementary active measurement to explore alternative protocol combinations. A separate question concerns the learning algorithm itself, including aspects such as result stability and predictability, particularly as network conditions change. Our exploratory analyses to date suggest that such learning is feasible, but much work remains to be done to understand how to learn for a set of protocols, and also how to combine multi-layer input.

- **Combinatorial explosion:** While intra-layer protocol choice seems manageable, it alone will not suffice. The performance gains of selecting a protocol at one layer might be impaired by the choice in another layer, suggesting a global optimization problem. However, a naive top-down approach immediately suffers from combinatorial explosion. Even modest numbers of alternatives (consider HTTP vs. HTTPS vs. HTTP/2, a small set of DNS resolvers, TCP vs. QUIC, IPv4 vs. IPv6, and WiFi vs. 3GPP connectivity) result in dozens of potential stacks that PolyStack would need to evaluate in order to produce a reliable global optimum. Conducting active measurements to explore this choice space seems challenging even when done opportunistically to benefit future connections.

- **Cross-layer coordination:** The combinatorial explosion problem also poses a technical challenge in terms of structuring cross-layer interaction. In order to evaluate the performance of a connection that uses a specific set of protocols, PolyStack needs to instruct each layer of the protocol stack to choose protocols where normally the layers would make such decisions autonomously. Indeed, such cross-layer coordination directly runs against the grain of core notions of protocol layering. We are of course not the first to make this observation, but believe that a holistically performance-aware stack exacerbates this problem.

- **Policy control:** Points might exist in the protocol choice space which the user explicitly does not want PolyStack to consider. For example, the user might find a particular DNS resolver undesirable, or veto the idea of reverting to insecure HTTP for performance gains. Today the user needs to configure desirable policies separately, across layers and systems (*e.g.* via DNS resolver configuration or use of

HTTPS-enforcing browser plugins). Comprehensive active measurement may also considerably increase the volume of traffic generated by the endpoint, which the user may find objectionable for example when resulting in costly additional cellular traffic. PolyStack does not necessarily lessen the user's control overall; instead, it exploits the policy-controlled freedom of choice more aggressively than current systems do.

- **Interaction with other optimizations:** In practice, complications may arise from the fact that PolyStack does not attempt to optimize performance in isolation. Such optimization conceptually already exists in the backbone (e.g., via performance-enhancing proxies) and on the server-side (e.g. via CDN technology), potentially rendering measurement tasks more challenging due to an increased number of moving parts. We argue that careful measurement can overcome such challenges, and advocate a paradigm in which client-side performance optimization complements, not collides, with existing optimizations elsewhere in the network. As we have outlined earlier in the paper, PolyStack's concepts do not only apply on the client, but operating on the end-host does maximize its effectiveness.

- **Performance vs. QoE:**

  In the above we have made an underlying assumption that better application performance (in terms of higher throughput or lower latencies) equals better QoE for the user. While we believe this assumption to hold in general, we acknowledge that it may oversimplify the problem setting. By allowing applications to control the stack's protocol composition, we hope to enable applications (and therefore also the user) to better control QoE in whatever form seems most suitable to a given application.

4 Summary

The strict layering of the TCP/IP protocol stack reduces the ability to leverage the rich protocol and technology diversity in today's networked systems. As a result, applications' performance and delivered QoE become suboptimal: it is unlikely that any protocol and network configuration, no matter how well designed, will outperform other protocols under *all* settings. In this paper we propose PolyStack, a performance-driven networking stack design that combines protocol diversity with the knowledge gathered through passive and active measurements to optimize QoE. PolyStack rests on the principles of layer-level transparency to existing implementations, global state available to all layers of the protocol stack, and extensibility in terms of inferring and improving performance to complement recent backbone- and server-focused approaches to improving application performance.