15-744 Final Project Proposal

Improving Tor Performance with Google's Quic

Kevin Ku (kku)
Xiaofan Li (xli2)

Introduction

Quic is Google's experimental transport layer protocol that supports multiplexing over UDP and provides security that's comparable to TLS. It supports ordered and reliable delivery, while avoiding the pain points of multiplexing over TCP. It also supports pluggable congestion control algorithms and synchronization cookie caching to reduce connection establishment time.

In their paper, "Performance and Security Improvements for Tor: A Survey," AlSabah and Goldberg outlined several issues that negatively impact Tor's performance. Out of all the issues, we found 2 that we think can be improved with Quic. First of all, the authors point out that Tor suffers from the head-of-line problem. Due to the fact that Tor multiplexes multiple circuits onto a single TCP connection, if the first packet is lost, TCP will buffer all the other packets belonging to different circuits until the first packet can be retransmitted. This significantly degrades performance since the packets belonging to different circuits don't depend on the packet that was lost. Secondly, they argue that Tor lacks a proper congestion control mechanism. The problem stems from TCP's poor support for multiplexing. Consider the situation where we have 2 circuits, one running file sharing and another running web browsing, multiplexed onto the same TCP connection. As the file sharing circuit drop more packets, TCP congestion control will kick in. Such congestion control will unfairly impair the performance of the web browsing circuit.

In addition, we think Quic can also reduce the communication cost in constructing a circuit in Tor. There are 2 aspects to this improvement. First of all, since each user and relay needs to establish TLS connection to another relay, Quic's ability to reuse synchronization cookie can reduce the latency for establishing these connections. Secondly, since the cost of circuit construction latency is reduced, users can establish new circuits more quickly. In turn, users can keep less backup circuits on standby, thereby reducing the circuit construction load on each relay.

In this project, we plan to evaluate the performance gains from replacing Tor's current TCP stack with Quic. Initially, we'll experiment with default Quic configuration, which solves the head-of-line problem. Next, we hope to play around with Quic's pluggable congestion control framework to find a good congestion control algorithm for Tor.

Challenges

The Tor project listed some concerns of using UDP instead of TCP. Some of the concerns, such as lack of good application-level stream support and encryption issues that arrive from packet drops, are solved by Quic. We are unsure of the following problems:

1. We may need to do IP-level packet normalization to guard against device fingerprinting attacks.
2. Certain protocols like DNS might leak information.
3. Tor namespace (".onion" addresses) might not be supported.
4. Exit policies on exit nodes for arbitrary IP packets need an IDS to enforce.

Despite the potential problems, we believe that they can all be resolved with more application logic. In addition, we aim to evaluate the performance gains of using Quic in Tor, while these problems mostly deal with anonymity of the service.

Plans

As shown in Figure 1, using the already established circuit, the client will connect to the guard relay from the OP (Onion Proxy). The OP is responsible for taking the payload, converting it into a cell according to the Tor network specification, encryption, wrapping it with a Quic packet header and finally sending it to the guard relay. Then the other relays in the network will do the normal decryption and forwarding of the cell on behalf of the client. However, instead of using TCP, they will communicate with the Quic protocol. At the exit relay, the payload is retrieved and sent to the conversing server with a normal TCP packet.

To perform our experiments, we need to set up 3 Tor relays plus an end host. We plan on running the relays on 3 different cloud providers and in 3 different geographical locations in order to simulate real network traffic. Therefore, it would be great if we could get some AWS and Google Cloud educational credit to run an instance on AWS.

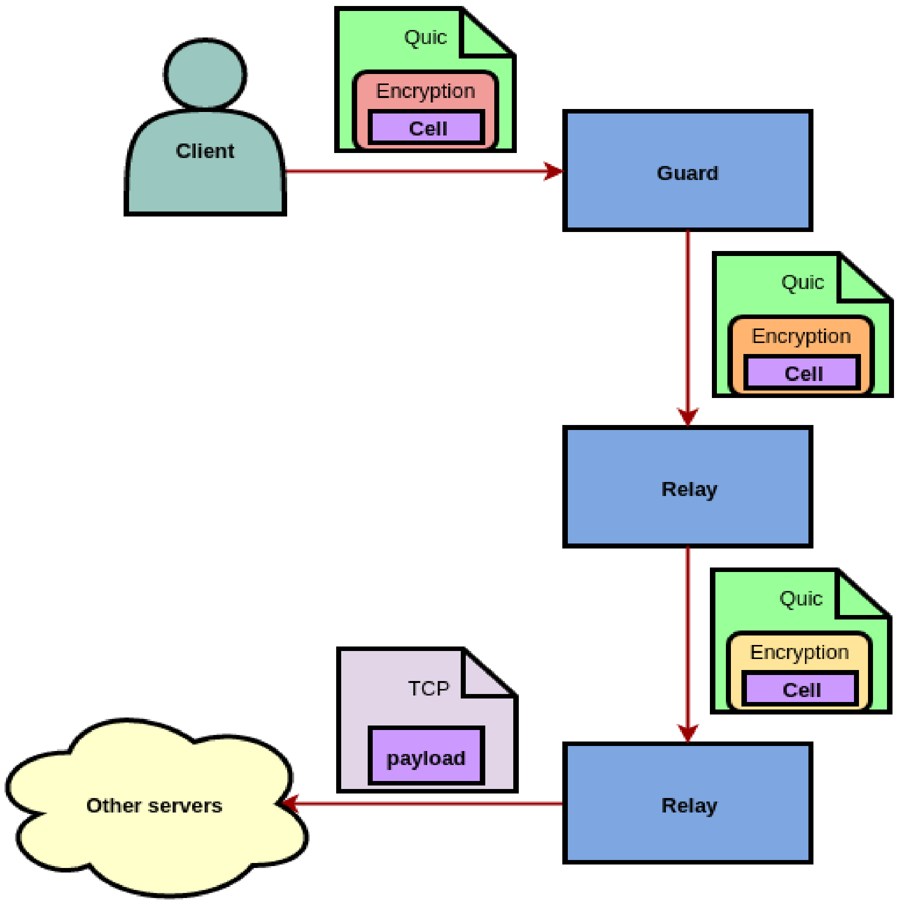

Figure 1: Concept Diagram of Tor Running on Quic