effectively raising or lowering the rate at which the sender can transmit. In TCP, this information is fed back to the sender in the `AdvertisedWindow` field of the ACK for every segment. One of the key issues in a rate-based protocol is how often the desired rate—which may change over time—is relayed back to the source: Is it for every packet, once per RTT, or only when the rate changes? While we have just now considered window versus rate in the context of flow control, it is an even more hotly contested issue in the context of congestion control, which we will discuss in the next chapter.

QUIC

QUIC originated at Google in 2012 and was subsequently developed as a proposed standard at the IETF. Unlike many other efforts to add to the set of transport protocols in the Internet, QUIC has achieved widespread deployment. As discussed further in Chapter 9, QUIC was motivated in large part by the challenges of matching the request/response semantics of HTTP to the stream-oriented nature of TCP. These issues have become more noticeable over time, due to factors such as the rise of high-latency wireless networks, the availability of multiple networks for a single device (e.g., Wi-Fi and cellular), and the increasing use of encrypted, authenticated connections on the Web (as discussed in Chapter 8). While a full description of QUIC is beyond our scope, some of the key design decisions are worth discussing.

If network latency is high—say 100 milliseconds or more—then a few RTTs can quickly add up to a visible annoyance for an end user. Establishing an HTTP session over TCP with Transport Layer Security (Section 8.5) would typically take at least three round trips (one for TCP session establishment and two for setting up the encryption parameters) before the first HTTP message could be sent. The designers of QUIC recognized that this delay—the direct result of a layered approach to protocol design—could be dramatically reduced if connection setup and the required security handshakes were combined and optimized for minimal round trips.

Multipath TCP

It isn't always necessary to define a new protocol if you find an existing protocol does not adequately serve a particular use case. Sometimes it's possible to make substantial changes in how an existing protocol is implemented, yet remain true to the original spec. Multipath TCP is an example of such a situation.

The idea of Multipath TCP is to steer packets over multiple paths through the Internet,

Note also how the presence of multiple network interfaces might affect the design. If your mobile phone loses its Wi-Fi connection and needs to switch to a cellular connection, that would typically require both a TCP timeout on one connection and a new series of handshakes on the other. Making the connection something that can persist over different network layer connections was another design goal for QUIC.

Finally, as noted above, the reliable byte stream model for TCP is a poor match to a Web page request, when many objects need to be fetched and page rendering could begin before they have all arrived. While one workaround for this would be to open multiple TCP connections in parallel, this approach (which was used in the early days of the Web) has its own set of drawbacks, notably on congestion control (see Chapter 6). Specifically, each connection runs its own congestion control loop, so the experience of congestion on one connection is not apparent to the other connections, and each connection tries to figure out the appropriate amount of bandwidth to consume on its own. QUIC introduced the idea of streams within a connection, so that objects could be delivered out-of-order while maintaining a holistic view of congestion across all the streams.

Interestingly, by the time QUIC emerged, many design decisions had been made that presented challenges for the deployment of a new transport protocol. Notably, many "middleboxes" such as NATs and firewalls (see Section 8.5) have enough understanding of the existing widespread transport protocols (TCP and UDP) that they can't be relied upon to pass

for example, by using two different IP addresses for one of the end-points. This can be especially helpful when delivering data to a mobile device that is connected to both Wi-Fi and the cellular network (and hence, has two unique IP addresses). Being wireless, both networks can experience significant packet-loss, so being able to use both to carry packets can dramatically improve the user experience. The key is for the receiving side of TCP to reconstruct the original, in-order byte stream before passing data up to the application, which remains unaware it is sitting on top of Multipath TCP. (This is in contrast to applications that purposely open two or more TCP connections to get better performance.)

As simple as Multipath TCP sounds, it is incredibly difficult to get right because it breaks many assumptions about how TCP flow control, in-order segment reassembly, and congestion control are implemented. We leave it as an exercise for the reader to explore these subtleties. Doing so is a great way to

a new transport protocol. As a result, QUIC actually rides on top of UDP. In other words, it is a transport protocol running on top of a transport protocol. This is not as uncommon as our focus on layering might suggest, as the next two subsections also illustrate. This choice was made in the interests of making QUIC deployable on the Internet as it exists, and has been quite successful.

make sure your basic understanding of TCP is sound.

QUIC implements fast connection establishment with encryption and authentication in the first RTT. It provides a connection identifier that persists across changes in the underlying network. It supports the multiplexing of several streams onto a single transport connection, to avoid the head-of-line blocking that may arise when a single packet is dropped while other useful data continues to arrive. And it preserves (and in some ways improves on) the congestion avoidance properties of TCP, an important aspect of transport protocols that we return to in Chapter 6.

HTTP went through a number of versions (1.0, 1.1, 2.0, discussed in Section 9.1) in an effort to map its requirements more cleanly onto the capabilities of TCP. With the arrival of QUIC, HTTP/3 is now able to leverage a transport layer that was explicitly designed to meet the application requirements of the Web.

QUIC is a most interesting development in the world of transport protocols. Many of the limitations of TCP have been known for decades, but QUIC represents one of the most successful efforts to date to stake out a different point in the design space. Because QUIC was inspired by experience with HTTP and the Web—which arose long after TCP was well established in the Internet—it presents a fascinating case study in the unforeseen consequences of layered designs and in the evolution of the Internet.