# Chapter 10

# Conclusions and Future Work

We begin this chapter by recapitulating the challenges of Web data transport and the solutions we have developed. We then discuss the contributions of this thesis. We re-visit the design principles laid out in Section 1.8, and evaluate to what extent we have conformed to them. Finally, we point out directions for future work, both in addressing some of the shortcomings of this work and in exploring new avenues.

## 10.1 Challenges and Solutions

This work has been motivated by the rapid and continued growth of the World Wide Web, and the performance problems that have arisen because of a mismatch between the characteristics of the Web application and the capabilities of the TCP protocol, the *de facto* standard protocol for reliable unicast data transport in the Internet.

Two characteristics of the Web application make it quite different from traditional applications, such as bulk data transfer. First, Web page data is composed of multiple distinct components, each in itself of some value to the user. So it is desirable to transfer the components concurrently and independently to minimize user-perceived latency. Second, data transfer happens in relatively short bursts, with long intervening idle periods.

These characteristics of the Web application cause a mismatch with TCP. There are two problems. First, there is a tight coupling between TCP's ordered byte-stream service model, and its congestion control and loss recovery mechanisms. So mapping each Web transfer to a separate TCP connection, as in HTTP/1.0, is inefficient. Second, it is difficult to utilize network bandwidth effectively when a transfer is short in length.

The objective of this thesis has been to resolve this mismatch while minimizing changes to the existing Internet protocol architecture. Specifically, our goals were to:

1. Provide efficient support for multiple, concurrent data streams between a server and a client.

2. Enable effective utilization of network bandwidth even when data transfer is short in length.

In addition, the growth in popularity of the Web has led to a proliferation of access network technologies. A new and growing subset of these are asymmetric networks, such as those based on cable modem technology. Improvements in fiber-optic and switching technologies promise to make the core of the network extremely fast, so it is increasingly more likely that access networks will play a critical role in determining end-to-end performance. So our third goal was to:

3. Enable efficient data transport over a wide range of access network technologies, including those with asymmetric bandwidth, large bandwidth-delay product, and very limited bandwidth.

We have developed several techniques to meet these goals. These include *persistent-connection HTTP*, *TCP session*, *TCP fast start*, and *TCP for asymmetric networks*. We discuss these techniques in the context of general paradigms presented in the next section.

## 10.2 Contributions

The contributions of this thesis are a set of general paradigms to evolve unicast data transport from the current state-of-the-art, and specific realizations of these that address the goals listed above. We discuss these paradigms next.

### 10.2.1 Separation of Protocol Service Model from Protocol Mechanisms

The service that a network protocol provides and the mechanisms used to support it should not be coupled together tightly in a one-to-one correspondence. In the context of TCP, the ordered byteservice provided by a TCP connection should not require that the congestion control and loss recovery procedures needed to support this service model also operate within the confines of a connection.

Our initial attempt in this direction resulted in *Persistent-Connection HTTP* (*P-HTTP*). The key idea in P-HTTP is sharing a single TCP connection to support multiple logical data streams, each providing a reliable, ordered byte-stream service (just like a TCP connection). The key ideas in P-HTTP have been adopted by the HTTP/1.1 protocol. However, P-HTTP has the limitation that it imposes an ordering across the logical data streams. This drawback led us to an alternative approach, which we discuss next.

### 10.2.2 Shared State

The resources of the Internet are shared by many users, hosts and applications. To achieve fair and effective utilization of the resources, the communicating entities should share among themselves the information they learn about the availability of these resources. Such sharing of information should be explicit, not be just implicit (as it is in TCP).

Our specific realization of this paradigm, in the context of communication between a pair of hosts (such as a Web server and a client), is *TCP session*. By integrating congestion control and loss recovery across multiple TCP connections, TCP session decouples these activities from the individual connections and instead uses shared information for the benefit of all.

This paradigm fits in particularly well in the context of multicast data transport. A good example is the use of join-experiments in Receiver-driven Layered Multicast [70] to share information about the level of congestion in the network across multiple receivers.

### 10.2.3 Persistent State

While the availability of resources in a shared network varies with time, information learned in the recent past may be useful in estimating availability of resources in the near future. Indeed, several studies (e.g., [7], [88]) have concluded that the availability of resources, such as bandwidth, in the Internet tends to remain quite stable for significant lengths of time.

We realize this paradigm, both in the context of an individual TCP connection and that of a TCP session, via the *TCP fast start* algorithm. By caching and reusing information such as the congestion window size and the round trip time estimate, TCP fast start enables even short transfers to utilize the network bandwidth effectively, thereby decreasing latency.

There is however, the possibility of the information learned in the past being rendered stale by significant changes in the network conditions. This requires that precautions be taken, which TCP fast start does, when reusing information from the past. Among the mechanisms employed by TCP fast start for this purpose is priority dropping, which we discuss next.

### 10.2.4 Exploiting Support for Differentiated Services

When appropriate, protocols and/or applications on end-hosts should exploit support for service differentiation that the network may provide. In the context of the Internet, there is an effort underway [25] to define light-weight mechanisms to support service differentiation at the granularity of packets. Such mechanisms provide the end-host protocols and applications the means to enforce control on a finer time-scale than a round trip time (which is what standard TCP is limited to).

We realize this paradigm in the context of TCP fast start by exploiting a *priority dropping* mechanism. This gives the sender the freedom to reuse information about the availability of network resources that it learned in the past while limiting possible adverse consequences in case this information is stale.

### 10.2.5 Redefining the Coupling Between the Data and Control Paths

A data transport protocol can be thought of as having two components — the *data path*, which is concerned with the actual process of sending data, and the *control path*, which determines when, how much and which data gets sent. The coupling between the data and control paths need to be defined with care.

In TCP, the flow of acks represents the control path, and its coupling with the flow of data is defined by the ack clocking mechanism. While this mechanism has several advantages, there are situations (as the investigations in this thesis have revealed) where it leads to performance limitations. For instance, dependence on ack clocking forces a connection to go through slow start to get the clock started. *TCP fast start* loosens this coupling by substituting software timers for the ack clocks during the first RTT of a connection. Software timers have been used to clock out data in transport protocols, such as NETBLT [22], that use rate-based flow control.

Another instance where the tight coupling between the flow of acks and the flow of data hurts is in asymmetric networks. The high degree of asymmetry in network characteristics in opposite directions means that the flow of acks often does not reflect a true picture of the network characteristics in the data path, so it should not control the flow of data. We have developed several techniques, including a*ck congestion control*, *ack filtering*, *acks-first scheduling*, and *sender adaptation*, to loosen the coupling between the two, thereby improving performance substantially.

### 10.2.6 Summary

In summary, by exploiting the paradigms discussed above, we have greatly enhanced the performance and flexibility of unicast data transport within the framework of TCP. Applications are free to use as many TCP connections as they wish without running the risk of degraded performance. Even data transfers that are short in length are able to utilize the network resources effectively. Good performance is obtained over a wide variety of access networks, including asymmetric ones.

Our contributions also include mathematical and trace-based performance analysis, simulation, implementation, and performance evaluation of the ideas developed in this thesis.

## 10.3 Design Principles Re-visited

We re-visit the design principles laid out in Section 1.8 to determine the extent to which we have succeeded in conforming to them.

### 10.3.1 Robustness

We believe that our work has benefited from being constructed on the robust TCP/IP protocol framework. In places where we have deviated from this framework, we have included specific mechanisms to guarantee robustness. For instance, TCP fast start uses the priority dropping mechanism to avoid penalizing other traffic in the network in case the cached information is stale.

Another example is the use of software timers to smooth out the sending of data when acks are infrequent, as in asymmetric-bandwidth networks.

### 10.3.2 Wide Applicability

The techniques we have developed are not tied to specific applications. Also, in large part, they are not tied to specific networking technologies. We believe these make the techniques applicable in a wide variety of situations.

The one possible exception is the set of techniques developed in Chapter 9. While these have focussed on asymmetric-bandwidth networks, they are also useful in combating other forms of asymmetry, such as that arising due to media-access characteristics. Furthermore, some of the techniques, such as sender adaptation, find uses in other contexts, such as TCP fast start, as well. Other mechanisms, such as ack congestion control, are triggered automatically only when there is congestion along the ack path. They are essentially harmless in other situations.

### 10.3.3 End-to-End Principle

The techniques we have developed are confined to hosts in the periphery of the network. The two exceptions are ack filtering and priority dropping, both of which require router support. However, ack filtering is confined to the upstream router in an asymmetric-bandwidth network. Also, since such networks tend to be at the periphery of the Internet rather than at its the core, the overhead on the router tends to be quite low.

The requirement for priority dropping may impact a larger fraction of the routers in the Internet. However, the mechanism itself is very simple. It does not require any state to be maintained or any complex computations to be performed in the routers. It just requires that packets be classified based on a single bit in the packet header. We believe that the robustness that this simple mechanism helps provide in the context of TCP fast start is well-worth its cost. And as we discuss next, this mechanism is being deployed for other reasons anyway.

### 10.3.4 Incremental Deployment

The operation of a majority of the techniques that we have developed are confined to the data sender. These include TCP session and much of TCP fast start. Confining operation to the sender facilitates deployment in the context of the Web, because only Web servers need to be modified and the (larger number of) clients are left untouched. Also, our experiments show that performance often improves even if the techniques are deployed only on a subset of the servers.

The main stumbling block to incremental deployment of TCP fast start is its dependence on the priority dropping mechanism. While such a mechanism is being deployed as part of the differentiated services effort [25], support for it is far from ubiquitous at this time. However, in certain situations, it is possible to derive much of the benefit of fast start with only a partial deployment, for instance, at a Web proxy located at a satellite earth station or at the headend of a cable network plant.

### 10.3.5 Summary

In summary, we believe we have been quite successful in conforming to the design principles laid out in Section 1.8. The shortcomings are mainly due to the dependence on router mechanisms such as priority dropping. While this does impede deployment to an extent, we believe that it also opens the door to a new paradigm for transport protocols, namely, exploiting differentiated services mechanisms in routers.

## 10.4 Directions for Future Work

We now discuss several directions for future work, both in addressing the limitations of this work and in exploring new avenues.

### 10.4.1 Sharing State Across Hosts

While TCP session shares state across connections between a pair of hosts, there are situations where it may be advantageous to extend such sharing across multiple hosts. For instance, client hosts on a local area network that are communicating with the same server may benefit from sharing information about the availability of network resources. Another example is a single client communicating with multiple server nodes in a cluster. There are several new issues that need to be addressed in this context — when is it appropriate to share state and on what time-scales should such sharing be done. The SPAND system [97] is an instance where such sharing is being investigated, albeit at the level of applications than the data transport protocol.

### 10.4.2 Comprehensive Use of Persistent State

TCP fast start adopts a very simple model for persistent state. To first order, it only reuses the most recent values of the congestion window size[1] and the round trip time estimate. It may be possible to improve the chances that a fast start attempt succeeds by making comprehensive use of information from the past. For instance, the evolution of the congestion window size over several round trip times in the past, suitably filtered, could be used to determine an appropriate setting for the congestion window. Another possibility, suggested by Mogul [75], is to consider the history of packet losses to decide whether to be aggressive or conservative in using persistent state.

### 10.4.3 Guarding Against Malicious Users

A shared network such as the Internet is vulnerable to malicious users who may appropriate an unfair share of the network resources by being aggressive in their use of the network. There have been proposals such as [32] to incorporate mechanisms in routers to detect and penalize such behavior. While not altering the situation fundamentally, the techniques that we have developed in this thesis pose new challenges to such policing. For instance, with TCP session, bundles of connections would need to be policed as a whole rather than each connection separately. The use of priority dropping as a protocol mechanism raises the question of whether different yardsticks should be used to police different classes of packets.

### 10.4.4 Improved Metrics for Quantifying Perceived Performance

Traditional network performance metrics such as throughput and latency, while useful, do not accurately capture the performance *perceived* by a human user. This is particularly so with multimedia data, as in the Web. Even a partially downloaded Web page may be useful from a user's

---

1. This is not strictly true. As discussed in Section 8.2.2.2, the old congestion window size may be modified before reuse.

viewpoint. And, for instance with progressively coded images, having the base layer of two images may be more valuable to a user than the having entire contents of one image and none of the other. So the figure of merit may well be how quickly the perceptually-important data is downloaded, not necessarily all of the data[2]. Quantifying perceived performance accurately is a difficult task. It requires accounting for several factors, including the perceived value of a partially downloaded object (the signal-to-noise ratio may serve as a crude measure in the case of images), the distinction between information that is visible to the user versus that which remains hidden, etc. There has been some initial effort in this direction, for instance, in [39] and [43].

### 10.4.5 Multicast-Based Dissemination of Web Data

There have been quite a few proposals for multicast-based dissemination of Web data (e.g., [104], [109]). The basic idea is to use multicast to avoid transmitting a given piece of Web data repeatedly through the network. While this is certainly desirable, there are several impediments to the general use of multicast in the context of the Web. These include the asynchronous nature of Web accesses, the updating of data at unpredictable times, and the limited overlap between accesses by different users (as evident from the relatively low hit rates of Web caches). Therefore, the interesting question is how multicast could be used to complement rather than replace unicast in a comprehensive architecture for disseminating Web data. One example is to dissipate hot spots caused by *flash crowds*, by marrying prefetching [87] with multicast-based data dissemination, as in the LSAM system [104].

## 10.5 Availability

The software we have developed and papers describing various parts of this work are available from `http://www.cs.berkeley.edu/~padmanab/phd-thesis.html`. The software includes our implementations of TCP session, TCP fast start, and the various techniques for alleviating the problems due to asymmetry. All of these implementation are part of the BSD/OS 3.0 TCP/IP stack. In addition, our user-level implementation of P-HTTP as part of the Mosaic and httpd programs from NCSA is also available.

---

2. As an aside, we note that connection scheduling in TCP session is well-suited to supporting a policy that, for instance, requires a larger fraction of the available bandwidth to be devoted to perceptually-important data.