The Web supports access to content stored on a cloud; virtually all cloud computing infrastructures

allow users to interact with their computations on the cloud using Web-based systems. Thus, it should

be clear that the metrics related to Web access are important for designing and tuning networks.

The site http://code.google.com/speed/articles/web-metrics.html provides

statistics about metrics such as size and number of resources; Table 7.1 summarizes these metrics.

The statistics are collected from a sample of several billion pages detected during Google’s crawl and

indexing pipeline.

Such statistics are useful for tuning the transport protocols to deliver optimal performance in terms

of latency and throughput. HTTP, the application protocol forWeb browsers, uses TCP, which supports

mechanisms to limit the amount of data transported over the network to avoid congestion. Metrics,

such as the average size of a page or the number of GET operations, are useful to explain the results

of performance measurements carried out on existing systems and to propose changes to optimize the

performance, as discussed next.

Another example illustrating the need for the networking infrastructure to adapt to new requirements

is the TCP initial congestionwindow. Before we analyze this problem in depth, we reviewtwo important

mechanisms to control data transport, called flow control and congestion control. TCP seeks to achieve

high channel utilization, avoid congestion, and, at the same time, ensure a fair sharing of the network

bandwidth.

TCP uses a sliding window flow control protocol. If W is the window size, then the data rate S of

the sender is:

S = W   MSS

RTT

bps = W

RTT

packets/second,

where MSS and RTT denote the maximum segment size and the round-trip time, respectively. If

S is too, small, the transmission rate is smaller than the channel capacity, whereas a large S leads to

congestion. The channel capacity in the case of communication over the Internet is not a fixed quantity,

but, as different physical channels are shared among many flows, it depends on the load of the network.

The actual window size W is affected by two factors: (a) the ability of the receiver to accept new

data and (b) the sender’s estimation of the available network capacity. The receiver specifies the amount

of additional data it is willing to accept in the receive window field of every frame; the receive window

shifts when the receiver receives and acknowledges a new segment of data. When a receiver advertises a

window size of zero, the sender stops sending data and starts the persist timer; this timer is used to avoid

the deadlock when a subsequent window size update from the receiver is lost. When the persist timer

expires, the sender sends a small packet and the receiver responds by sending another acknowledgment

containing the new window size. In addition to the flow control provided by the receiver, the sender

attempts to infer the available network capacity and to avoid overloading the network. The source uses

the losses and the delay to determine the level of congestion. If awnd denotes the receiver window and

cwnd the congestion window set by the sender, the actual window should be:

W = min (cwnd, awnd).

Several algorithms are used to calculate cwnd, including Tahoe and Reno, developed by Jacobson in 1988

and 1990. Tahoe was based on slow-start (SS), congestion avoidance (CA), and fast retransmit (FR); the

sender probes the network for spare capacity and detects congestion based on loss. The slow start means

that the sender starts with a window of two times MSS, init\_cwnd = 1; for every packet acknowledged,

the congestion window increases by 1MSS so that the congestion window effectively doubles for every

RTT. When the congestion window exceeds the threshold, cwnd   ssthresh, the algorithm enters the

congestion avoidance state; in CA state, on each successful acknowledgment cwnd ← cwnd+1/cwnd

and on each RTT cwnd ← cwnd + 1. The fast retransmit is motivated by the fact that the time out is

too long, so a sender retransmits immediately after three duplicate acknowledgments without waiting for a timeout; two adjustments are then made:

flightsize = min (awnd, cwnd) and ssthresh ← max (flightsize/2, 2)

and the system enters in the slow-start state, cwnd = 1.

The pseudocode describing the Tahoe algorithm is:

for every ACK {

if (W < ssthresh) then W++ (SS)

else W += 1/W (CA)

}

for every loss {

ssthresh = W/2

W = 1

}

The pattern of usage of the Internet has changed. Measurements reported by different sources [108]

show that in 2009 the average bandwidth of an Internet connection was 1.7 Mbps; more than 50% of

the traffic required more than 2 Mbps and could be considered broadband, whereas only about 5% of

the flows required less that 256 Kbps and could be considered narrowband. Recall that the averageWeb

page size is in the range of 384 KB.

Although the majority of Internet traffic is due to long-lived, bulk data transfer (e.g., video streaming

and audio streaming), the majority of transactions are short lived (e.g., Web requests). So a major

challenge is to ensure some fairness for short-lived transactions.

To overcome the limitations of the slow-start application, strategies have been developed to reduce

the time to download data over the Internet. For example, two browsers, Firefox 3 and Google Chrome,

open up to six TCP connections per domain to increase the parallelism and to boost startup performance

in downloading aWeb page. Internet Explorer 8 opens 180 connections. Clearly, these strategies circumvent

the mechanisms for congestion control and incur a considerable overhead. It is argued that a better

solution is to increase the initial congestion window of TCP, and the arguments presented in [108] are:

• The TCP latency is dominated by the number of RTTs during the slow-start phase4; increasing the

init\_cwnd parameter allows the data transfer to be completed with fewer RTTs.

• Given that the average page size is 384 KB, a single TCP connection requires multiple RTTs to

download a single page.

• It ensures fairness between short-lived transactions that are the majority of Internet transfers and the

long-lived transactions that transfer very large amounts of data.

• It allows faster recovery after losses through fast retransmission.

In the experiments reported in [108] the TCP latency was reduced from about 490 msec when

init\_cwnd = 3 to about 466 msec for init\_cwnd = 16.