October 05, 2016

"METHOD FOR IMPROVED BANDWIDTH UTILIZATION IN DATA DOWNLOADING SYSTEMS USING INTELLIGENT DYNAMIC CONNECTION LIMIT STEPPING"

Simon Hatch

Oystein Eftevaag

METHOD FOR IMPROVED BANDWIDTH UTILIZATION IN DATA DOWNLOADING SYSTEMS USING INTELLIGENT DYNAMIC CONNECTION LIMIT STEPPING

Any piece of computer software running on a client computing device that needs to download multiple pieces of data (or data files) from a network (e.g., a web browser downloading multiple images and cascading style sheet (CSS) files for a web page), would need to throttle the number of concurrent files it downloads due to bandwidth limitations. If the client computing device attempts to download too much concurrently, the resources would likely contend with each other for bandwidth and individually take an extended amount of time to download. If the client computing device has too few concurrent downloads, the client computing device may not be utilizing its bandwidth optimally. The client computing device may include a mobile device, a wireless device, a stationary desktop device, a multimedia device, a set-top box device, a streaming device, a television device, or any other client capable of requesting and downloading data from a server.

Browsers and other systems that need to download multiple files and/or resources typically set a fixed number of total connections per host (referred to as slots), which represents an approximation of that balance between bandwidth utilization and concurrent downloads for most situations. This approach however has its drawbacks thereby making this approach less optimal. This approach results in overwhelming the connection in very bandwidth-constrained situations, and underutilizing the connection on high-bandwidth connections.

Another drawback with this approach when it comes to web browsers, however, becomes apparent when the client computing device needs to download a large number of small resources. Every file that a client computing device downloads comes with some time overhead before the client computing device can begin receiving actual data: The client computing device would need to initiate a TCP/IP connection, potentially perform secure sockets layer (SSL) handshaking, and then request the resource (See Figure 1). This can mean several roundtrips from the client computing device to the server, which takes time (some multiple of the latency to send a signal between the two endpoints).

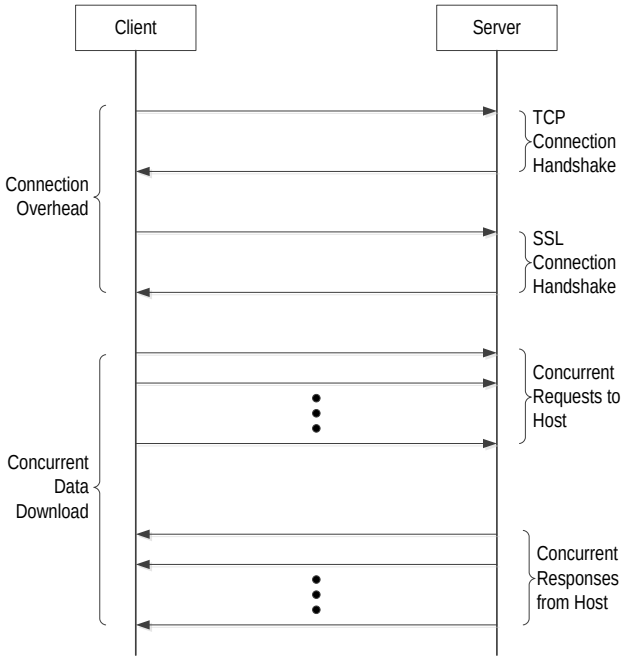

Figure 1 – Client/Server Round Trip Exchanges

The end result of this round trip exchange is that each time the client computing device starts a download and utilizes a slot, there is a time period where the connection in that slot is not actually using up any bandwidth at all since the slot is simply waiting for a response with data from the server. In a situation where the client computing device has a large number of relatively small resources, the overall bandwidth utilization would therefore become relatively

small and the client computing device may end up spending more time on connection overhead than actual data transmission. In another situation, the available bandwidth for mobile devices can frequently and suddenly change, which means attempts by the mobile device to consume the actual bandwidth by the conventional approaches outlined above as a mechanism to govern the number of used connections may be challenging. Finding the right balance between the bandwidth utilization and number of concurrent downloads may be desirable.

To improve the bandwidth utilization in data download systems, the connection limit (or the number of slots) is dynamically set by a heuristic algorithm, which monitors the state of the active connections and steps up or down the allowed number of connections based on what overall percentage of the active connections are currently in a waiting-for-data state (i.e., connection overhead). In stepping through the connection limit, the client computing device may allow for some slack in bandwidth to be available in order to compensate for sudden state changes while staying within bandwidth utilization limits. The process of connection limit stepping may be as follows:

- Start with a default number of connections to a host (e.g., a default value of 6).
- Client computing device navigates to page. This page may be an initial resource obtained by the client computing device that indicates multiple sub-resources that need to be download by the client computing device to fully render the requested resource.
- Client computing device sends resource requests up to a predetermined maximum value. The client computing device can queue subsequent resource requests if maximum value already reached.

- As resources finish being download to the client, the client computing device may then send new resource requests. In this example, the client computing device may begin sending those subsequent resource requests in the queue.
- Client computing device calculates percentage of time spent on a connection as follows:

$$Time\ Spent\ \% = \frac{T_{DNS} + T_{SSL} + T_{TCP}}{T_{TOTAL}}$$

, where T_{DNS} is the time spent in DNS resolve, T_{SSL} is the time spent in SSL handshake, T_{TCP} is the time spent in TCP slow start, and T_{TOTAL} is the total time spent downloading (including time spent receiving data from the host). The percentage of time spent ratio may differ than shown above to account for other connection overhead delays (e.g., IPSec handshake), or may exclude some of the round-trip delays shown above, depending on implementation.

  - Client computing device records and stores history of time spent ratios for prior resources. The client computing device may log historical data relating to the prior ratios in a local cache or locally-accessible data repository.
- Client computing device utilizes a sliding window of most recent downloads. The time spent of downloads are averaged together, where time spent for recent downloads are weighted more heavily.
- If the percentage of time spent exceeds a predetermined threshold X, then the client computing device allows the allotted number of connections to increase up to a predefined minimum number of connections (e.g., min(default_max, current + 1)).

- If the percentage of time spent does not exceed a predetermined threshold Y, then the client computing device allows the allotted number of connections to decrease down to a predefined maximum number of connections (e.g., a max(default, current - 1)).

As listed above, the client computing device would start out with a default number of connections (or slots) per host, but at some time interval since initiating the connections, the algorithm allows the client computing device an opportunity to adjust the number of slots, up to predefined minimum and maximum number of connections (as a safeguard against sudden changes in circumstances; i.e. starting out by downloading many tiny resources, and suddenly switching to downloading some large resources).

If the client computing device calculates that the bandwidth in the connections is being under-utilized by determining that a large number of the active connections are waiting to receive data, then the client computing device increases the slot allotment so long as the adjusted number of connections remains under the maximum cap (See Figure 2).

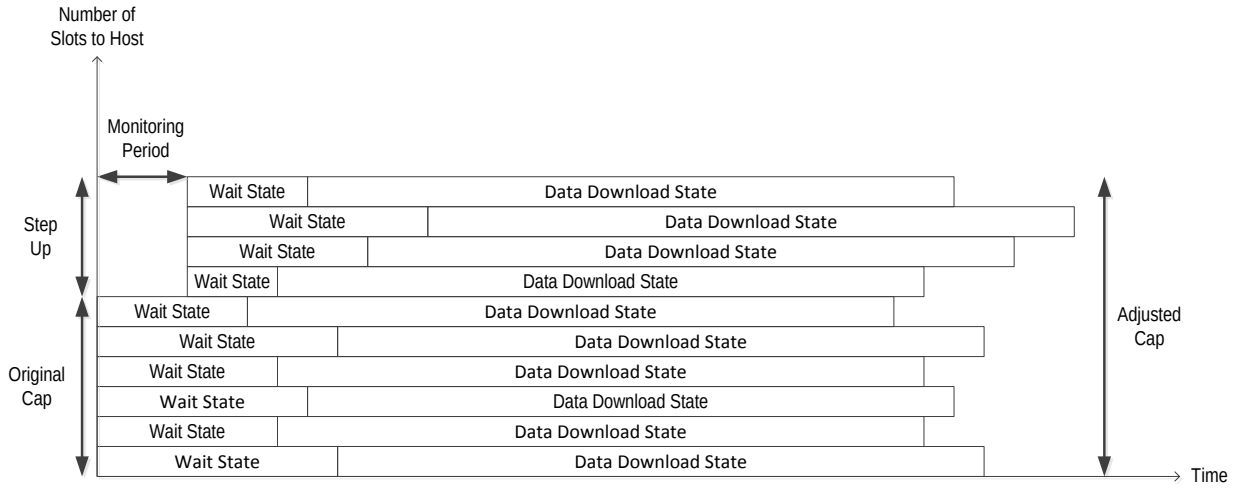

Figure 2 – Slot Allotment Increase

If the client computing device determines that a large number of the active connections are all receiving or sending data, the client computing device concludes that the bandwidth in the connections is being over-utilized, and hence, the client computing device decreases the slot allotment so long as the adjusted number of connections remains above the minimum cap (See Figure 3). In some aspects, the minimum cap is set to at least one connection.

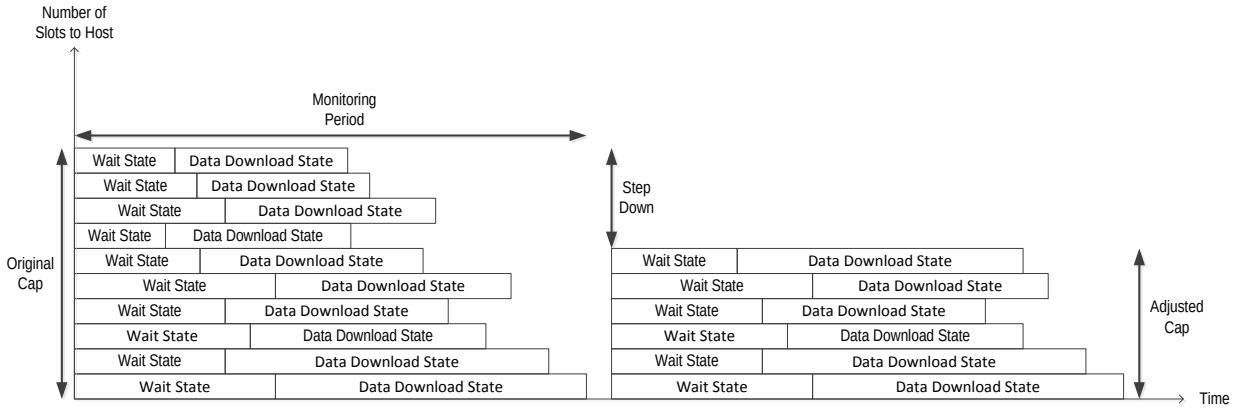

Figure 3 – Slot Allotment Decrease

The approach outlined above provides for the client computing device to obtain information to know whether a connection is under-utilizing or over-utilizing the available bandwidth by using the connection state instead. For example, the client computing device may automatically adapt to both the available bandwidth and to the size of the resources we need to download. If the client computing device is over-utilizing bandwidth over the concurrent connections, then the latency monitored is higher, and hence, a number of active connections will more often be in the waiting state. There is a direct correlation between the size of the resource and the overhead in connection time to download that resource. In other words, as the resource increases in size, the connection time becomes relatively smaller.

An alternative approach would be to experimentally increase/decrease the number of available slots and then monitor the actual bandwidth usage of the client computing device to see if the

change resulted in increased or decreased bandwidth utilization. Another approach would be to log historical data and establish the number of connections based on the historical data. The client computing device may store the historical data in a local cache or a locally accessible data repository. For example, once a web browser has visited a certain page, the client computing device retrieves the ratios for small to large resources, and the client computing device can use those historical ratios to configure the number of slots in advance (i.e., pre-configuration).

The process of connection limit stepping may be implemented by a browser vendor or an entity that develops a system that needs to download multiple pieces of data and/or files from across the network. The use of the dynamic connection limit stepping may be detectible by using a network monitoring tool to monitor how many concurrent connections a browser (or other system) is utilizing, under different circumstances. A set of webpages may be constructed with differently sized sub-resources (e.g., some pages with many small resources, some pages with many large resources). A browser using the dynamic connection limit stepping may be dynamically adjusting the number of TCP/IP connections over time as the page load progresses, depending on which site is being loaded.

ABSTRACT

The subject technology relates to improving bandwidth utilization of data downloading systems by dynamically setting a connection limit with a host. The connection limit (or number of slots) can be dynamically set by a heuristic algorithm that monitors the state of active connections with the host, and steps up or down the allowed number of connections based on what overall percentage of the active connections are currently in a waiting-for-data state. The dynamic connection limit stepping utilizes the connection state in order to automatically adapt to both the available bandwidth and the size of the resources needed to download.