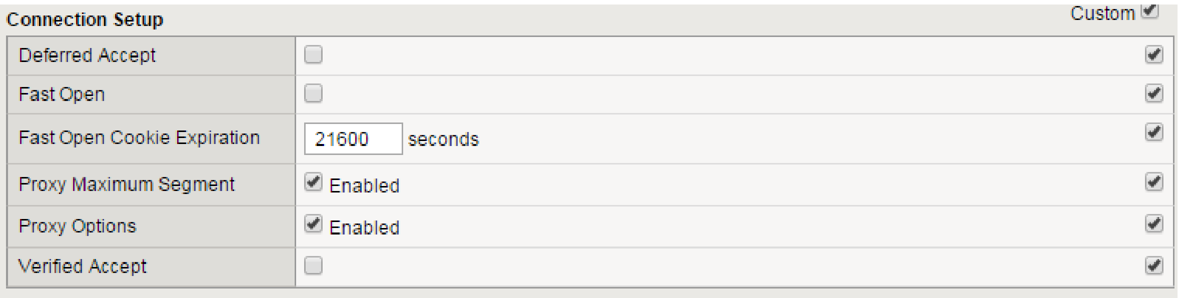
Tuning the TCP Profile, Part Two

Martin Duke, 2016-26-02

Last month I walked through the first two segments of the TCP profile configuration page. This month I'll continue this tour. As I said then, when I refer to "performance" below, I'm referring to the speed at which your customer gets her data. Performance can also refer to the scalability of your application delivery due to CPU and memory limitations, and when that's what I mean, I'll say so.

Connection Setup



Connection Setup		Custom ☑
Deferred Accept	☐	☑
Fast Open	☐	☑
Fast Open Cookie Expiration	21600 seconds	☑
Proxy Maximum Segment	☑ Enabled	☑
Proxy Options	☑ Enabled	☑
Verified Accept	☐	☑

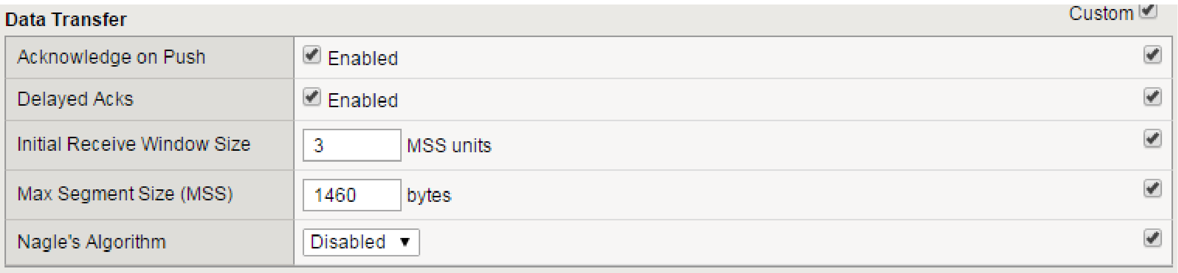
The big performance enhancer here is **Fast Open.** I discussed this in great detail earlier, and it can make a big difference for short connections. **Fast Open Cookie Expiration** specifies how long the cookie that the BIG-IP provides to a client is valid. Set it too short, and most Fast Open attempts will end up failing due to an expired cookie. Set it too long, and that's more time for a valid cookie to end up in the wrong hands. There is no memory impact on the BIG-IP in any case.

Proxy Maximum Segment helps out the CPU a bit without having any impact on your throughput under most conditions. When the **serverside** profile enables this, the serverside will not advertise an MSS that results in TCP data payloads that can't fit in a clientside MSS. This saves the effort of rewriting packets into separate memory. It accounts for options that might take up space on one side but not the other.

Proxy Options and **Verified Accept** are about signaling across the proxy. Proxy Options, when selected on the serverside profile, won't request timestamps on the serverside if it's not enabled on the clientside. Enable this if your servers are interested in this information. Verified Accept won't respond to a client SYN packet until it receives a SYN/ACK from the server. Thus your clients will (accurately) get a non-response from the server rather than a SYN/ACK followed by a TCP Reset. This might be important for security scans on your servers. On the other hand, this delays additional connection data from the client. Also note that Verified Accept changes the way that iRule events work, so use it with care.

Lastly, there is **Deferred Accept**, which is a defense against DoS attacks. BIG-IP won't set up anything above TCP on the clientside, or anything at all on the serverside, until the client completes the three-way-handshake. The reduces the per-connection overhead during a SYN flood, but once again slow things up on the serverside.

Data Transfer



Data Transfer		Custom ☑
Acknowledge on Push	☑ Enabled	☑
Delayed Acks	☑ Enabled	☑
Initial Receive Window Size	3 MSS units	☑
Max Segment Size (MSS)	1460 bytes	☑
Nagle's Algorithm	Disabled ▼	☑

Delayed Acks have been allowed by the TCP standard for a long time (RFC 1122). If your remote hosts are concerned about packet counts, possibly due to power constraints, they might appreciate fewer ACK packets. On the other hand, as the name implies, delayed ACKs can introduce delay into the data/ack control loop, slowing delivery of additional data. Furthermore, some congestion control algorithms will behave less aggressively if they receive fewer ACKs, even if that represents the same amount of data. So if you don't really care about packet counts, disable this.

Clients or servers might set the PSH flag on data packets to encourage an acknowledgment. **Acknowledge on Push** respects that request, even if BIG-IP would normally delay the ACK. It might cause your system to send more ACKs, but can prevent it from waiting around for additional data that isn't going to come. If you disable Delayed Acks, this option doesn't matter.

Nagle's Algorithm also relates to the tradeoff between packet counts and more conventional measures of performance. I discussed this option in detail last year.

The **Initial Receive Window Size** controls how much data BIG-IP will allow from the remote host before the first pure ACK packet from BIG-IP. Although setting it too large may consume more memory on your system, in general you will certainly want this to be large enough to accept the initial message from the client, whether it be an SSL message, HTTP request, or something else. For really large objects, you'll be limited by the remote host's initial congestion window or send buffer anyway, so a really large initial receive window size has diminishing returns. We limit it to 16 segments so that you can't get too crazy.

The **Max Segment Size** should be the path Maximum Transmission Unit (MTU) minus the minimum-size IP and TCP headers (40 bytes), unless your network usually uses IP header options. If you set this too low, more of the bytes you move will be per-packet overhead rather than application data. Set it too high, and in many cases routers will drop the first data you send to the remote host.

That's it for now. Next month we'll finish up with the next two blocks of significant performance-affecting sections, and a brief note on other security settings and MPTCP.