Questions on TCP FAST OPEN

Pankaj More

Y9227402

The average size of a web page today is around 300 KB. Why then is the average Page Load Time so large?

1. Average web object is around 7 KB. A single web page contains large number of web objects.
2. Not all web objects can be concurrently received in a single RTT due to many reasons(eg. data dependencies).

Why reducing the number of RTTs is the most effective way for improving the Page Load Time?

1. Bandwidth does not matter much since size of web page is small
2. Page Load Time depends on 2 things.

i. Round Trip Time - Already close to the achievable limit imposed by the speed of light(cannot do anything about it)
ii. Number of RTTs - Can be reducing using various techniques and scope for much improvement is possible.

What is the problem with HTTP Persistent Connections?

1. In general, host and middle-boxes terminate idle HTTP connections to minimize resource usage.
2. The middle-box issue may be partly mitigated by using TCP keep-alive probes, but this could be power hungry on mobile devices.

What is the problem with allowing data exchange before TCP Handshake is complete?

1. vulnerable to DOS attacks
2. may not work well with applications where duplicate or stale SYNs can cause problems

How does TCP Fast Open prevent source-address spoofing attack?

1. By using a security "cookie" for authenticating the client
2. A client that wishes to use TFO requests a cookie from the server in a regular TCP connection with the TFO TCP option included and uses that cookie to perform fast open in subsequent connections to the same server.

Give 2 examples of new security threats introduced by TCP Fast Open?

1. Server Resource Exhaustion
2. Amplified Reflection Attack

How does TCP Fast Open deal with the issue of Server Resource Exhaustion?

By maintaining a counter of total pending TFO connection requests which have not been migrated to fully-established TCP state. When the counter exceeds a predetermined threshold, server temporarily disable TFO and incoming TFO requests fall back on regular 3WHS. This allows usual SYN flood defense techniques to prevent further damage until the counter falls below the limit.

Give any 3 reasons why server performance increase on using TCP Fast Open even though it introduces additional overhead of cookie generation/encryption?

1. one RTT saved per request
2. fewer CPU cycles spent by the server to process each request(as the request is received in the SYN packet itself)
3. the AES encryption function is hardware-accelerated in modern processors and accounts for only 0.3 % extra overhead.

Why does TCP Fast Open not allow data packets to be sent by the client following a SYN packet?

1. Additional data packets would have the ACK flag unset since the initial sequence number of the server is unknown until the receipt of the SYN-ACK.
2. Most of the middle-boxes drop data packets without the ACK flag.
3. Hence, TCP Fast Open limits the amount of data sent by the client during 3WHS to a single MSS.

From a deploy-ability perspective, comment on how TCP Fast Open deals with the issue of NAT?

1. If NAT grants same IP to the client for all TCP connections, it can reuse the cookie.
2. If NAT gives different IP addresses for new TCP connections, the TFO cookies cached by the client will no more be valid. The server would reject the data in SYN packet and fall back on a regular 3WHS. However, this would not cause any latency penalty versus an ordinary TCP connection since the server would reply with a SYN-ACK.