Network support for TCP Fast Open

Christoph Paasch <cpaasch@apple.com>
• TCP Fast Open allows to reduce latency and significantly improve user-experience

• However, naive firewalls and bad Intrusion Detection Systems got in our way

We should change that!
Latency matters

PLT of 25 most popular websites (Latency = 60ms)

Page Load Time [ms]

Latency matters

PLT of 25 most popular websites (Latency = 60ms)

<table>
<thead>
<tr>
<th>Bandwidth [Mbps]</th>
<th>Page Load Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>750</td>
</tr>
<tr>
<td>2</td>
<td>4000</td>
</tr>
</tbody>
</table>

PLT of 25 most popular websites (Bandwidth = 5Mbps)

<table>
<thead>
<tr>
<th>Latency [ms]</th>
<th>Page Load Time [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>750</td>
</tr>
<tr>
<td>20</td>
<td>4000</td>
</tr>
</tbody>
</table>

Latency matters

• [2] measured impact of latency on service revenue

• Direct correlation between latency and revenue:
  ‣ 100ms of additional delay has significant impact on the revenue and customer satisfaction

Transmission Control Protocol

• Used for 95% of the Internet’s traffic
• Provides a reliable and in-order byte-stream service
• 3-way handshake to establish the connection
The TCP joke

- “Hi, I’d like to hear a TCP joke.”

- “Hello, would you like to hear a TCP joke?”

- “Yes, I’d like to hear a TCP joke.”

- “Ok, I’ll tell you a TCP joke.”

- ...
TCP Handshake is expensive
TCP Handshake is expensive

Cellular Network’s RTT can range in the hundreds of milliseconds
TCP Fast Open
(RFC 7413)

Accelerating the TCP Handshake
TCP Fast Open (TFO)

- Allows clients to send SYN with data
- Enables servers to reply right away with the response
- Protects itself against DoS through a cookie, unique for each client-IP
- Standardized at the IETF - RFC 7413

TFO at Apple

- TCP Fast Open in iOS 9 and OS X 10.11 (and later)
- Used for an Apple Service on all iOS and OS X devices
- Public API by using `connectx(2)`
- Overall, very beneficial

But, Firewalls got in our way
TFO in details

1. Cookie Exchange

Client

Server

SYN
TFO Cookie Request

Generate Cookie:
\[ X = \text{Hash} (\text{Client}_\text{IP}, \text{secret}) \]

SYN/ACK
TFO Cookie = X
TFO in details

2. Sending SYN + data

Client

SYN + data
TFO Cookie = X

Server

:Verify Cookie:
:X == Hash(Client_IP, secret)

SYN/ACK
TFO in details

3. Server replies with data

Client

Server

SYN + data
TFO Cookie = X

Verify Cookie:
X == Hash(Client_IP, secret)

SYN/ACK

Data

ACK
Middlebox issues with TCP Fast Open

... and their negative impact
Middlebox issues

- Bad Middleboxes and Firewalls respond badly to TCP Fast Open
  - **Suppress** TCP options
  - **Drop** packets
  - Mark entire connection as “invalid”
  - **Blackhole** the clients
Using a new TCP option

Issue

Simplistic middleboxes *remove* unknown TCP options
Using a new TCP option

**Issue**

Simplistic middleboxes **remove** unknown TCP options

Client

SYN

TFO Cookie Request

Server

SYN

SYN/ACK
Using a new TCP option

*Issue*

Simplistic middleboxes *remove* unknown TCP options

*Impact*

Clients *cannot use TFO*, and thus pay a latency cost compared to well-behaving networks
Using a new TCP option

**Issue**

Simplistic middleboxes drop segments with unknown TCP options
Using a new TCP option

Issue

Simplistic middleboxes drop segments with unknown TCP options

Client

RTO (1s) { SYN TFO Cookie Request }

Server

SYN
Using a new TCP option

**Issue**

Simplistic middleboxes drop segments with unknown TCP options

**Impact**

Client has to retransmit the SYN-segment without the TCP option. The user experiences a high page-load-time.
Sending SYN+data

*Issue*

Naive middleboxes *drop SYN segments* with data
Sending SYN+data

*Issue*

Naive middleboxes **drop SYN segments** with data

[Diagram showing communication flow between client and server with RTO, SYN, and SYN + data packets.]
Sending SYN+data

**Issue**

Naive middleboxes drop SYN segments with data

**Impact**

Clients has to retransmit the SYN-segment without the TCP option. The user experiences a high page-load-time.
Acknowledging SYN+data

*Issue*

The server **acknowledges the SYN+data**, thus more than the initial sequence number. Middleboxes might **drop the SYN/ACK**.
Acknowledging SYN+data

**Issue**

The server *acknowledges the SYN+data*, thus more than the initial sequence number. Middleboxes might drop the SYN/ACK.

Client

```
SYN (Seq=0, len=10) + data
```

Expected Ack=1

Server

```
SYN/ACK (Ack=11)
SYN/ACK (Ack=11)
```

Synack
Acknowledging SYN+data

**Issue**

The server **acknowledges the SYN+data**, thus more than the initial sequence number. Middleboxes might **drop the SYN/ACK**.

**Impact**

The middlebox keeps on blocking the server’s SYN/ACK. The session **never becomes established**.
Server sends data right before 3-way handshake completes

*Issue*

Bad Intrusion Detection Systems (IDS) start **blackholing** the client
Server sends data right before 3-way handshake completes

*Issue*

Bad Intrusion Detection Systems (IDS) start **blackholing** the client

Client  
- SYN + data  
- TFO Cookie = X  
- SYN/ACK  
- Data  
- ACK

Server
Server sends data right before 3-way handshake completes

*Issue*

Bad Intrusion Detection Systems (IDS) start **blackholing** the client

*Impact*

Client **loses connectivity** to the server. Subsequent connections (non-TFO) also might be blocked by the IDS.
How common is this?

Mostly, TFO works successfully (~80% success-rate).

But…
How common is this?

Mostly, TFO works successfully (~80% success-rate).

But…

100% of the users of the affected networks are penalized
Conclusion

• Latency has a direct impact on user-experience

• TCP Fast Open allows to significantly reduce latency

• Bad middleboxes are interfering with TCP Fast Open

Vendors and operators:

Take TFO into account for a better user-experience
References

Backup-slides
TFO and idempotency

- Data sent in a SYN might reach the server twice

TCP-session terminates

SYN + data got delayed in the network

Old SYN + data finally reaches the server
TFO and idempotency

• Use TFO only with “idempotent” data (aka., data that can be received twice by the server)

E.g.,:

• TLS (ClientHello)

• HTTP-Requests