Transports advancements in the Windows network stack

Praveen Balasubramanian

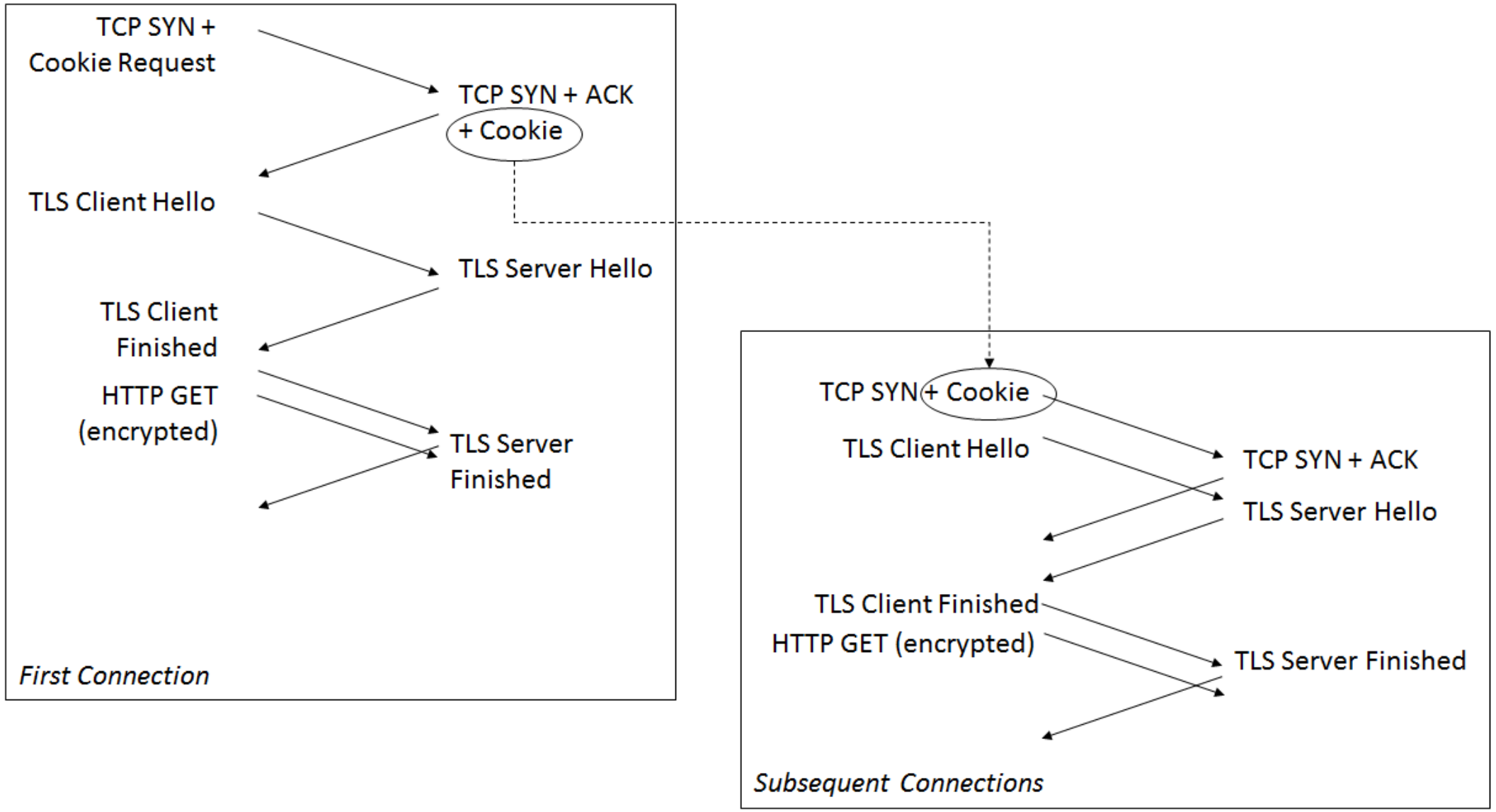
Engineering Lead, Data Transports & Security

pravb@microsoft.com

The fight against TCP ossification

- Key barriers to TCP innovation
  1. Slow OS updates
  2. Middleboxes
  3. Slow adoption of new features by services
- Continuous rapid updates – address problem 1
- Enable by default with safe fallbacks – partly address problem 2
- Fast moving services – address problem 3
- Windows 10 is now on 350 million+ systems
  - PC / tablet / mobile / Xbox / Hololens / IoT
  - 96 percent of major enterprise customers now piloting or deploying
- “Anniversary update for Windows 10” – free update for all users
- Windows Server 2016

TCP Fast Open (and TLS False Start)



TCP Fast Open

- First step towards 0-RTT (Microsoft is very interested in TLS 1.3)
- RFC 7413 compliant
- TCP global setting is enabled by default
- about:flags setting (off by default) in Microsoft Edge in Windows Insider Preview builds 14352 and higher
- Found interop issues with
  - Firewalls and anti-virus software
    - Mitigated by bug fixes in OS and firewall software code
    - RFC could include this as source of problems
  - Middleboxes
    - Same class of issues as found by Apple in iOS and OSX tests*
    - Requires detection and mitigation
    - Request tcpm community to standardize – maybe errata to RFC 7413?

*https://www.ietf.org/proceedings/94/slides/slides-94-tcpm-13.pdf

TCP Fast Open details

- IANA assigned option number 34
- State machine based approach (modeled as sub-states before estab)
- Stop negotiation after first SYN timeout
- No correlation between attempting TFO and retransmitting first SYN for top 10000 websites
- Most server properties enable TFO on port 443
- Client will accept server cookie of all valid sizes
- Server side support is in the works
- Sample code to enable and use TFO:

```
char option = 1;
int rc = setsockopt(sock, IPPROTO_TCP, TCP_FASTOPEN, &option,
sizeof(option));
// Use ConnectEx, an existing Winsock API that takes a send data // buffer
https://msdn.microsoft.com/en-us/library/windows/desktop/ms737606(v=vs.85).aspx
// If server accepted SYN with cookie request or cookie + data, // option
will be 1
int rc = getsockopt(sock, IPPROTO_TCP, TCP_FASTOPEN, &option, &optlen);
```

Initial Congestion Window 10 (IW10)

- IW limits amount of data in first RTT
- Up until Windows 10 and Server 2012 R2, the default IW = 4 MSS
- Anniversary update for Windows 10 and Server 2016, default IW = 10 MSS
- Loss in first burst, reset to IW = 4 MSS for the connection
- Initial telemetry shows increased loss in initial burst across device types. Working on
  - Quantifying the increased loss rate
  - A/B testing with IW = 4 and IW = 10
- SIO_TCP_SET_ICW socket option on client allows (2, 4 ,10 MSS). On server allows up to 64 MSS

Tail Loss Probe (TLP)

- Based on draft-dukkipati-tcpm-tcp-loss-probe-01
- Try to convert RTOs into Fast Retransmits
- Limit to only one probe segment
- WCDelAckT = 200 msec seems too conservative for single outstanding segment case
  - What value does the Linux kernel use?
  - Does Linux kernel have dynamic delayed ACK timeout tuning?
- On by default on Windows 10 Anniversary update and Server 2016
- TLP enabled only for connections that have an RTT of at least 10 msec in both client and server
- FACK with a fixed reordering threshold of 3 to improve TLP based recovery
- FACK is always enabled independent of TLP

Recent ACKnowledgement (RACK)

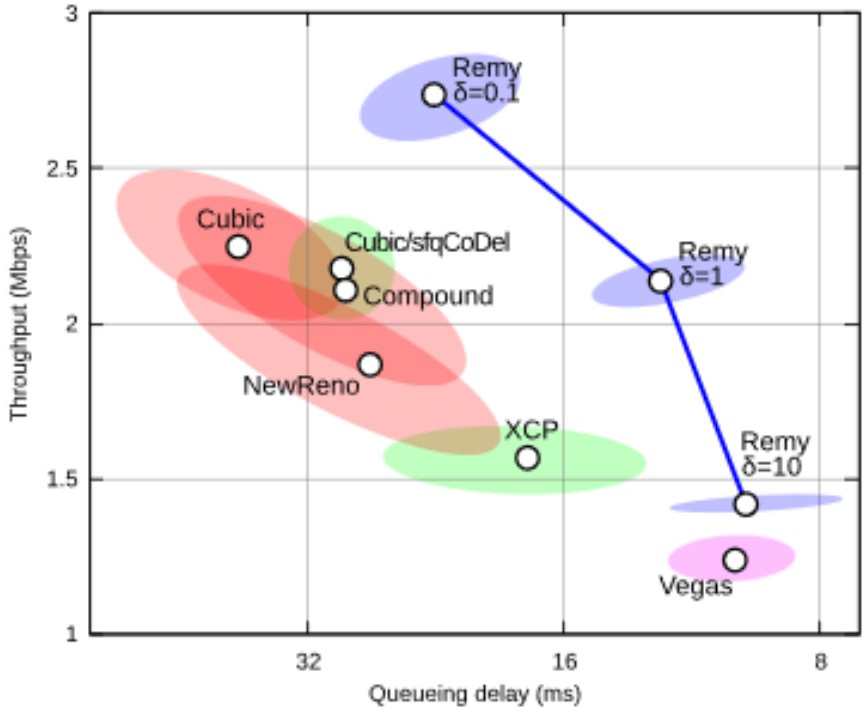
- Based on draft-cheng-tcpm-rack-00
- Use rotating buffer of “time slots” instead of time per send buffer
  - Tradeoff between precision and memory space
  - “Reading index” tracks forward with ACKs, “Insertion index” tracks new sends
  - LSO sends tracked in single slot – not expensive to convert LSO to standard send in recovery retransmissions
- On by default on Windows 10 Anniversary update and Server 2016
- RACK enabled only for connections that have an RTT of at least 10 msec in both client and server
- RACK.reo_wnd = min_RTT/4
- Increase in spurious RTOs with RACK enabled – investigating further
- Encourage tcpm to publish this as experimental RFC

LEDBAT

- The background downloads problem
- Prototype implementation based on RFC 6817
- Several proprietary deviations as a result of measurements that show sub-optimal behavior of stock LEDBAT
- Undocumented socket option – not ready for public consumption
- Under evaluation by several first party workloads
- More on this in the next IETF

Diversity or convergence of congestion control?

- Windows client → Compound TCP
- Windows Server → dynamic based on handshake RTT – DCTCP or Compound TCP
- Linux kernel → CUBIC. We are evaluating a prototype (not in shipping code)
- High throughput vs. queueing delay. Loss / ECN / delay machine learning.



- http://web.mit.edu/remy/
- http://web.mit.edu/remy/TCPexMachina.pdf

Q&A