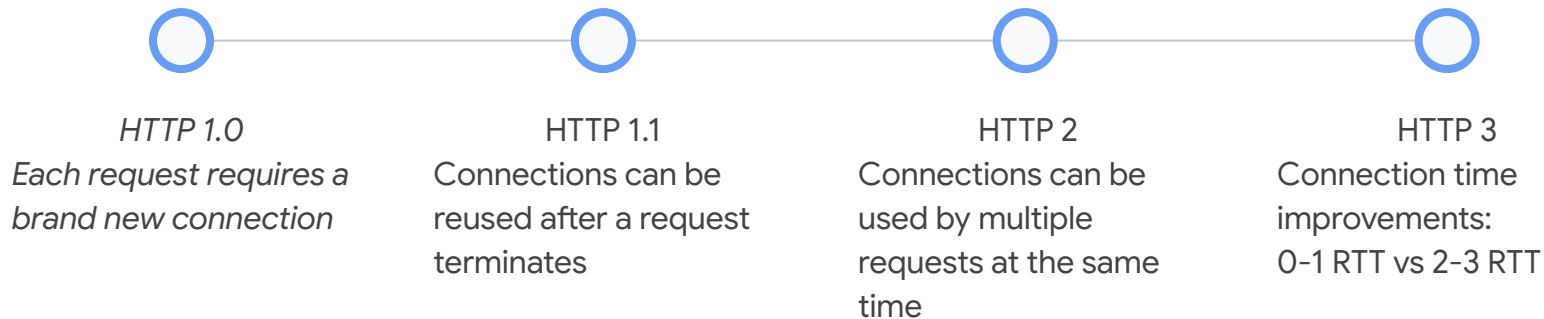
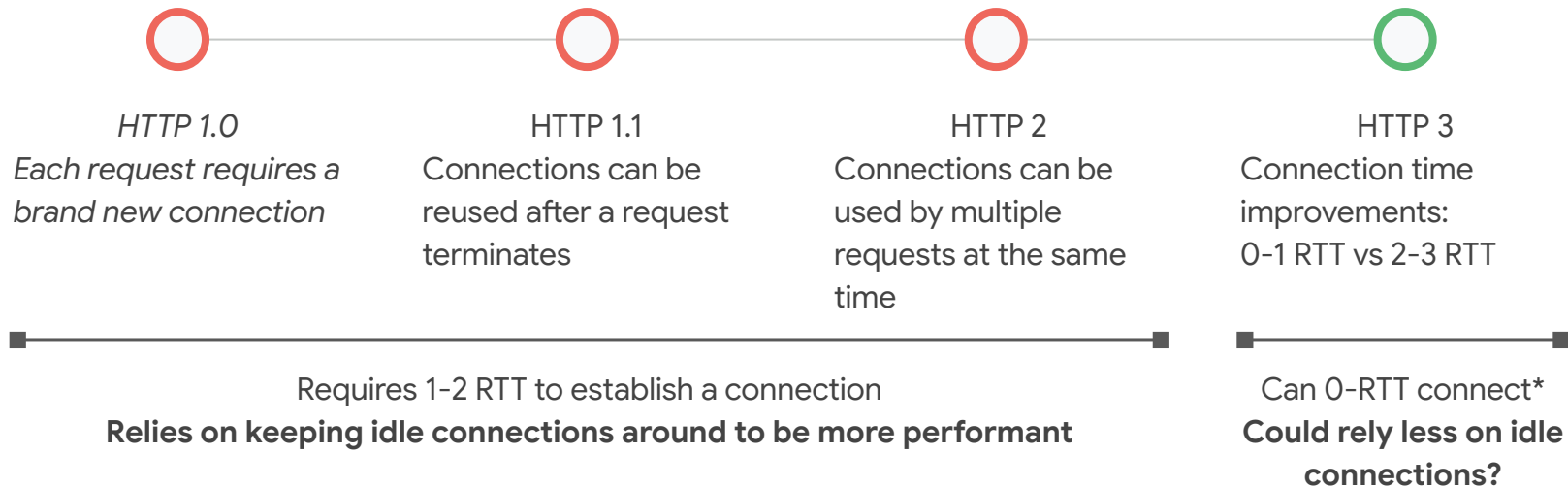


Idle connections & mobile: beneficial or harmful?

Connection establishment and HTTP requests



Connection establishment and HTTP requests



Flavors of 0-RTT

- [RFC 8446](#) - TLS 1.3
 - Support for 0-RTT Data
 - 0-RTT and [Anti-Replay](#)
 - Recommends accepting 0-RTT data at most once
 - Warns about operational costs of doing so
- [RFC 8470](#) - Using Early Data in HTTP
 - Allows sending HTTP data via TLS early data
 - Leaves [solving anti-replay](#) to clients
- [RFC 9000](#) - QUIC: A UDP-Based Multiplexed and Secure Transport
 - Allows sending application data before receiving a server response
 - Again, [leaves solving anti-replay to clients](#)
 - Suffers from [address validation](#)

Idle connections

- Idle connections provide pooling benefits only if:
 - Future requests can be coalesced into an idle connection (*speculative*)
 - Idle connection don't "break". This is done either via
 - Periodic pings (*can consume lots of energy, especially on mobile*)
 - Idle timeout tuning (*easy for middleboxes to drop your NAT entry*)
- Chrome studied the performance benefit of QUIC 0-RTT
 - Data presented at [IETF 115](#)
 - Results not as good as expected. Attributed to
 - Bugs and missing optimizations
 - Browser preconnects
 - Connection coalescing

What if "true 0-RTT" is a way out of idle connections?

"True & Safe 0-RTT"

- [RFC 9308](#) - Applicability of the QUIC Transport Protocol
 - \geq "true 0-RTT" is qualitatively different from the point of view of the client
 - Again, it just warns about replay attacks
- **Could we provide an HTTP construct to perform "safe 0-RTT"?**
 - Naive: a nonce HTTP header/TLS extension?
 - It does not need to support every scenario
 - We could gracefully fallback to 1-RTT if expensive to support
 - e.g. client moves and hits a different server