Networking Best Practices

Foundations for reliable and performant networking

Session 706

Joshua Graessley
Senior Software Engineer, Core Networking

“For every ailment under the sun,
There is a remedy, or there is none,
If there be one, try to find it;
If there be none, never mind it.”

Mother Goose

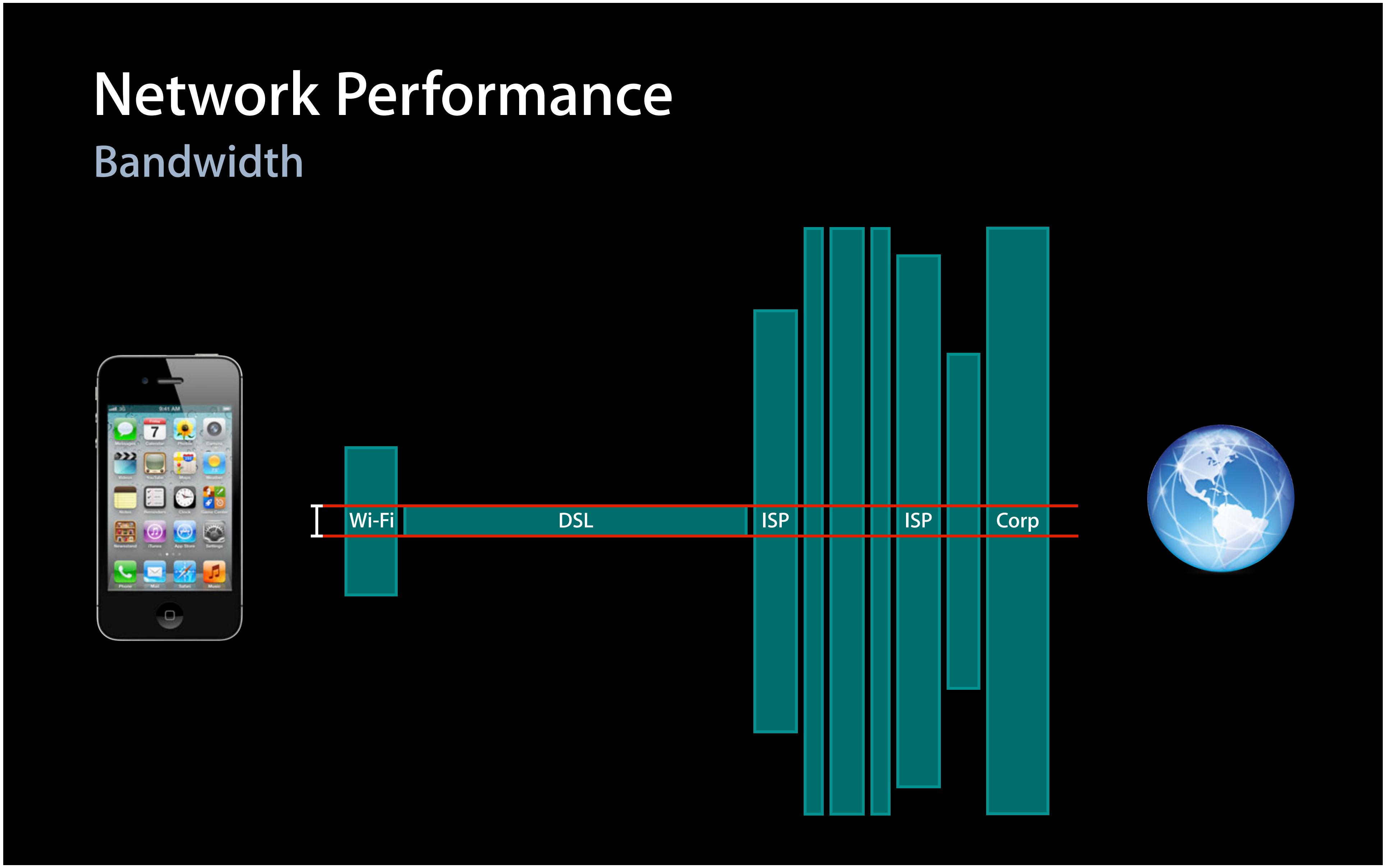
Abstractions

- Powerful
    - Hide complexity
    - Layered functionality
- Leaky
    - Hide true cost
- Best practices
    - Make the most of the abstractions
    - Pick the right layer

Overview

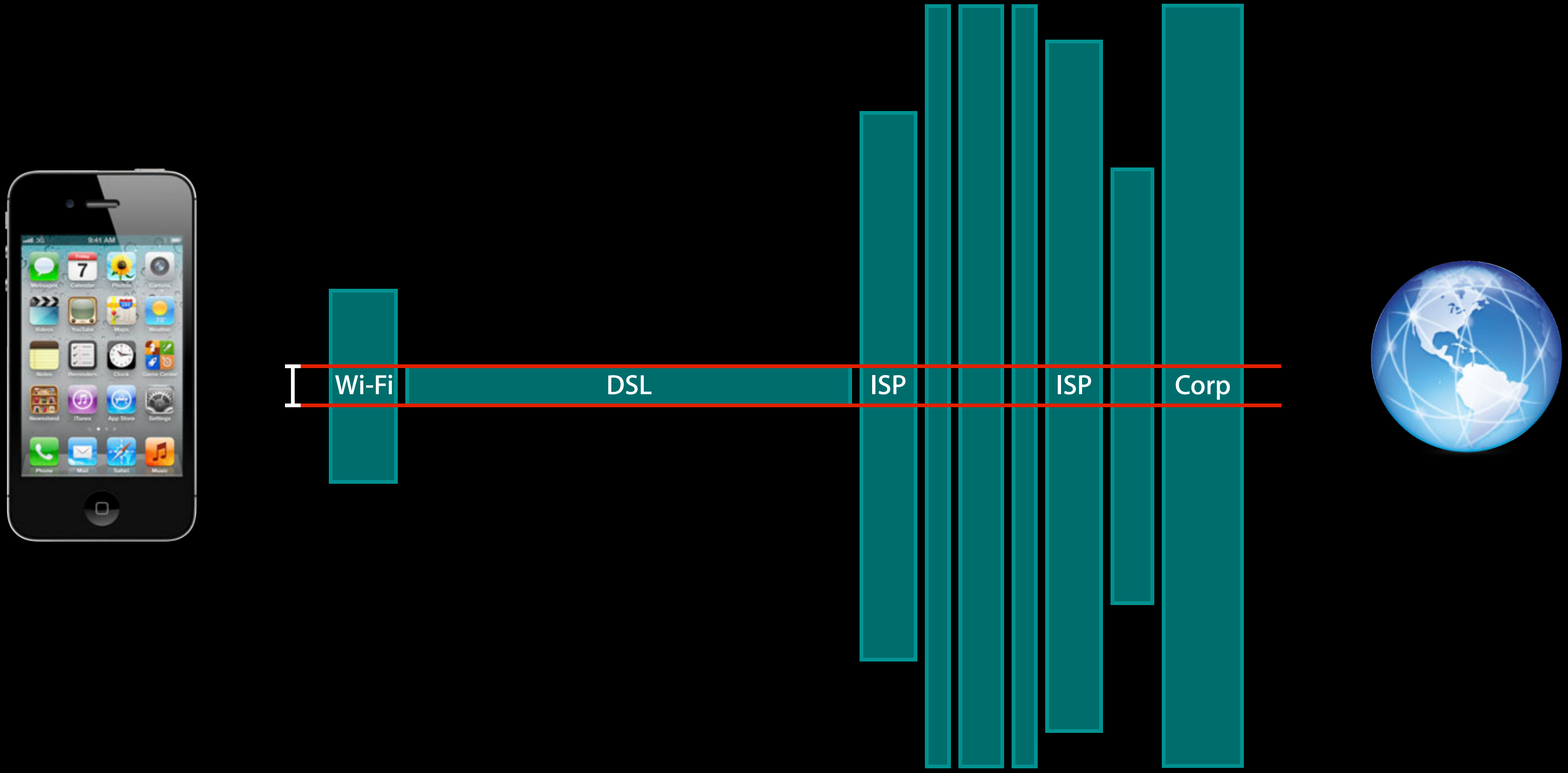
- Network performance
- Protocol abstractions
  - TCP
  - HTTP
- API abstractions
  - CFSocketStream
  - NSURLConnection
  - WebKit
- Mobility and cost
- Debugging problems

Network Performance



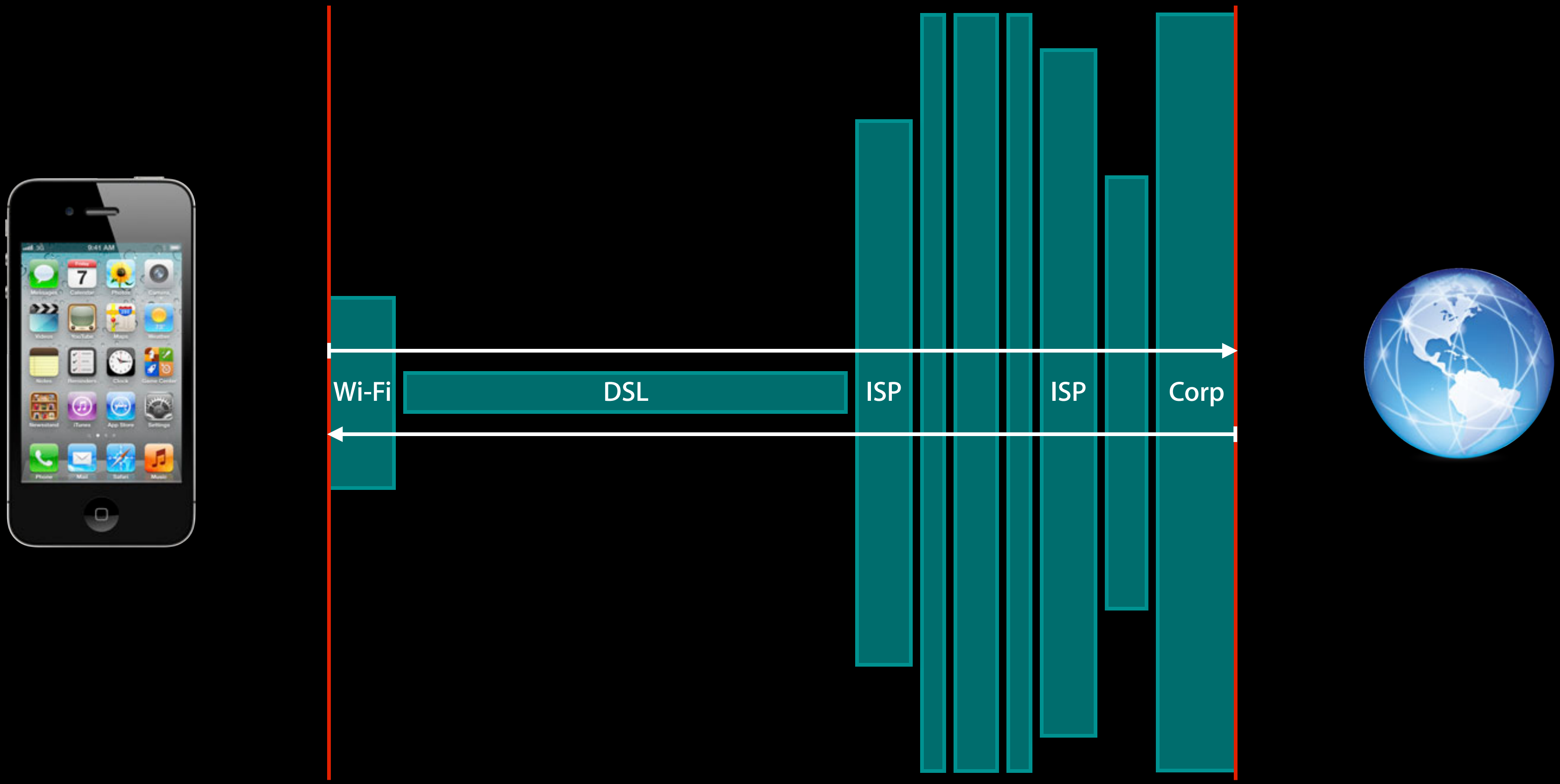
Network Performance

Bandwidth



Network Performance

Latency



Network Performance

Hiding latency

- Asynchronous networking
  - Responsive user interface
  - Placeholders
    - Fill in when data arrives
- One connection, concurrent requests
  - HTTP pipelining
- Request early
- Cache

Network Performance

Bandwidth delay product

60ms RTT = 16 round trips/sec = 25000 bytes/sec = 200 kbit/sec

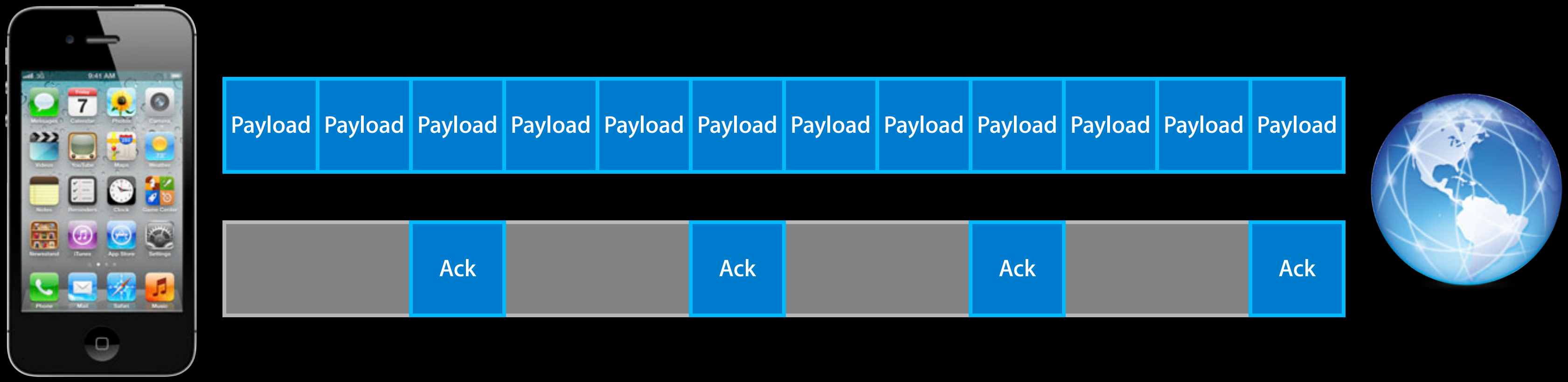
Network Performance

Bandwidth delay product

60ms RTT = 16 round trips/sec = 25000 bytes/sec = 200 kbit/sec

Network Performance
Bandwidth delay product

10megabit/sec * 60ms = 600kilobit = 75kilobytes = 50 packets



Network Performance

Summary

- Bandwidth = min (y)
  - Only bottleneck link matters
- Latency = sum (x)
  - Every link increases delay
  - Hide latency for big wins
- Bandwidth delay product
  - Bandwidth * RTT
  - Minimum in-flight data for peak throughput

Protocol Abstractions

Transmission Control Protocol (TCP)

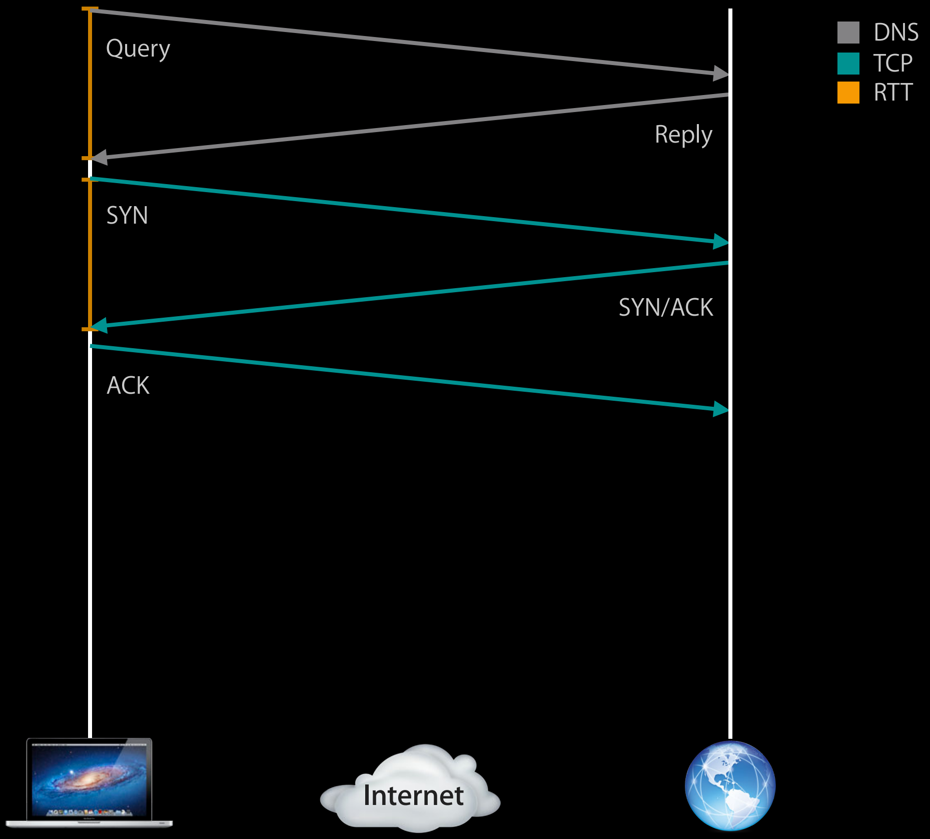
Transmission Control Protocol

Services provided

- Virtual circuit
  - Bi-directional serial byte stream
  - Reliable transmission
  - In-order delivery
  - Data integrity
  - Flow control
- Shared state at endpoints

Transmission Control Protocol

Three-way handshake



Transmission Control Protocol

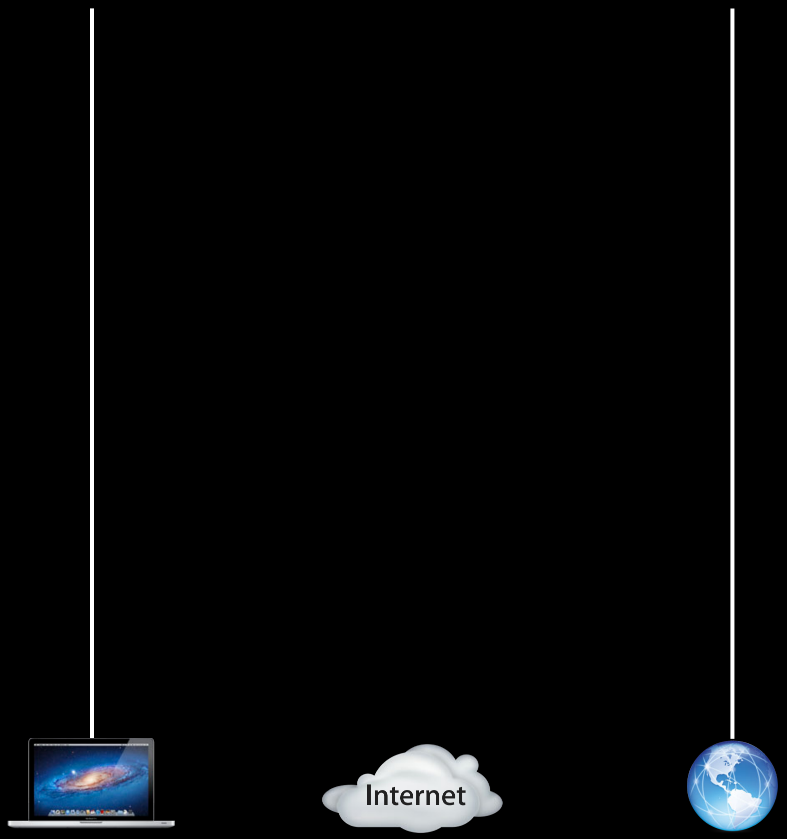
Sequence numbers

- Packet indicates sequence + length
  - Sequence = 1, length = 12
- In-order deliver
- Detect missing data
- Acknowledge received data
- Pseudo-random initial sequence number
  - tcpdump displays relative

1	2	3	4	5	6	7	8	9	10	11	12
H	e	l	l	o	_	W	o	r	l	d	!

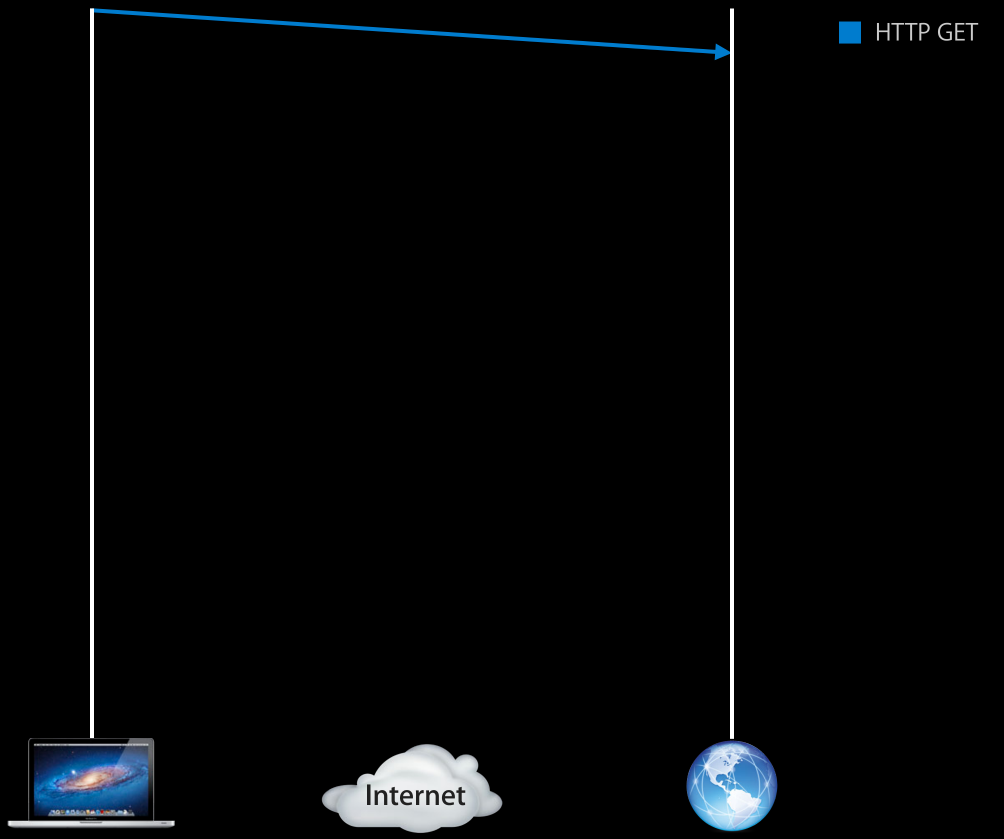
Transmission Control Protocol

Slow start



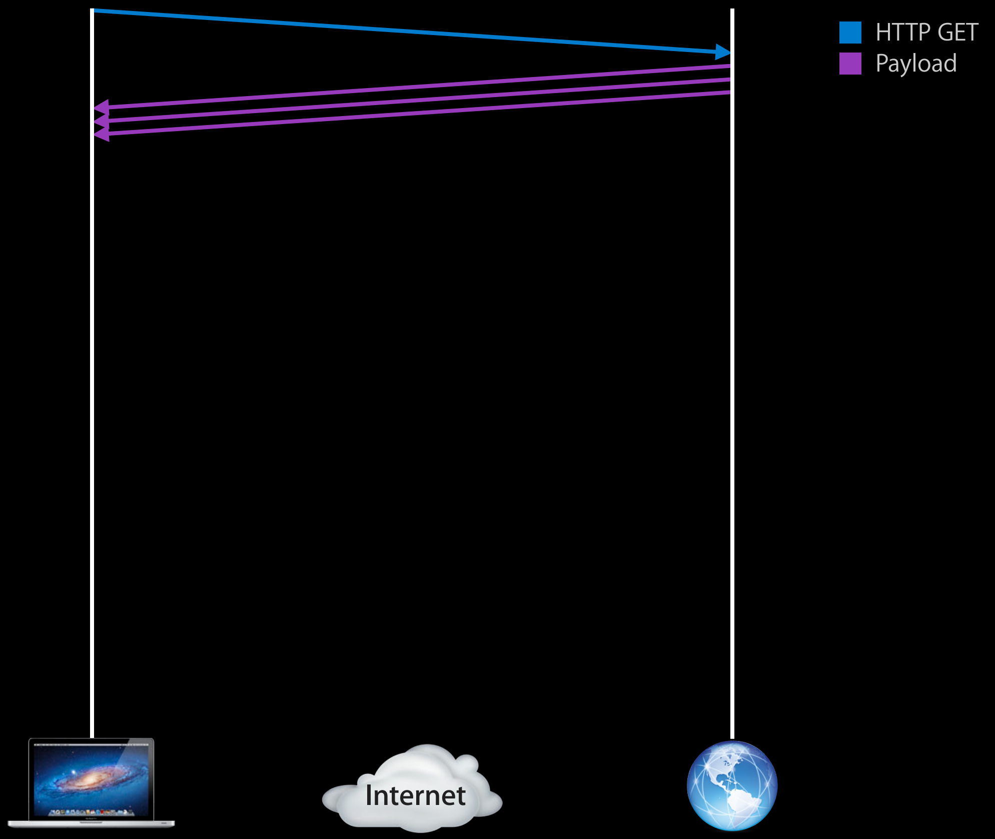
Transmission Control Protocol

Slow start



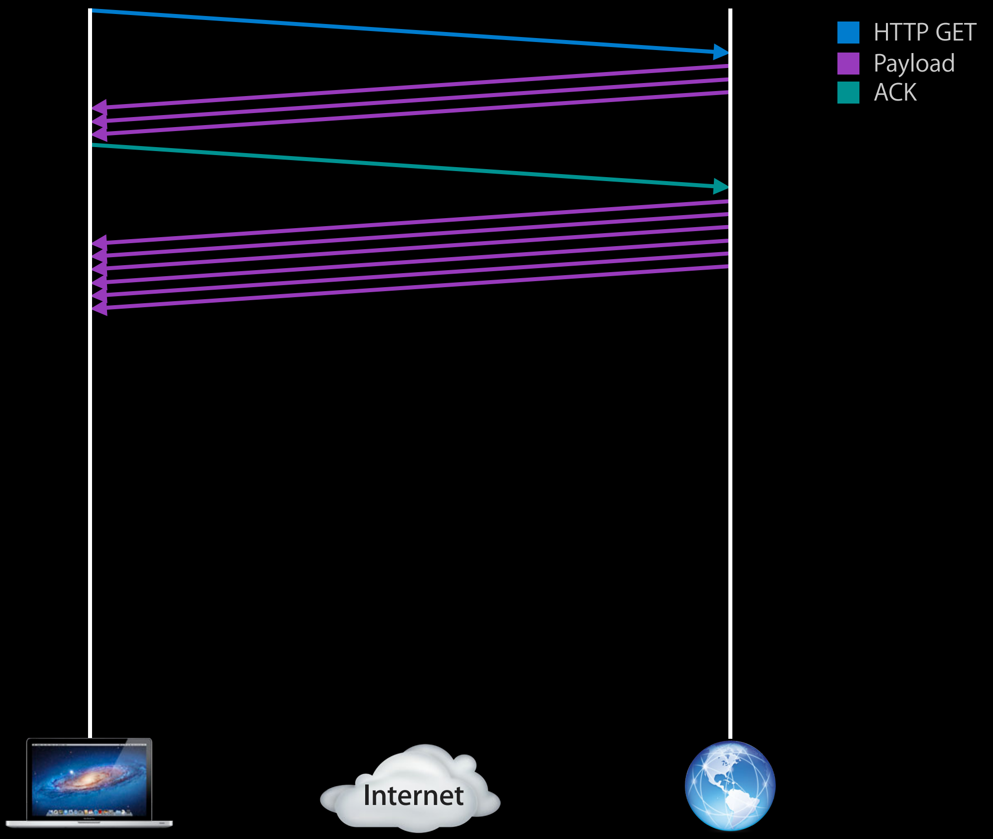
Transmission Control Protocol

Slow start



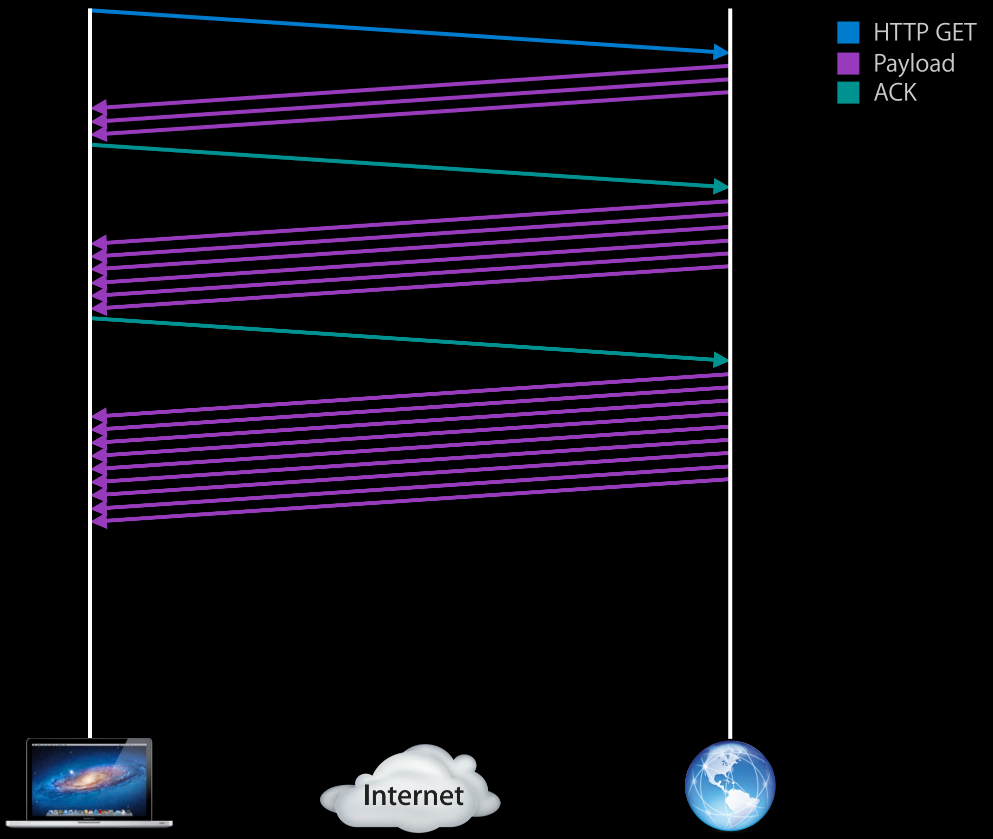
Transmission Control Protocol

Slow start



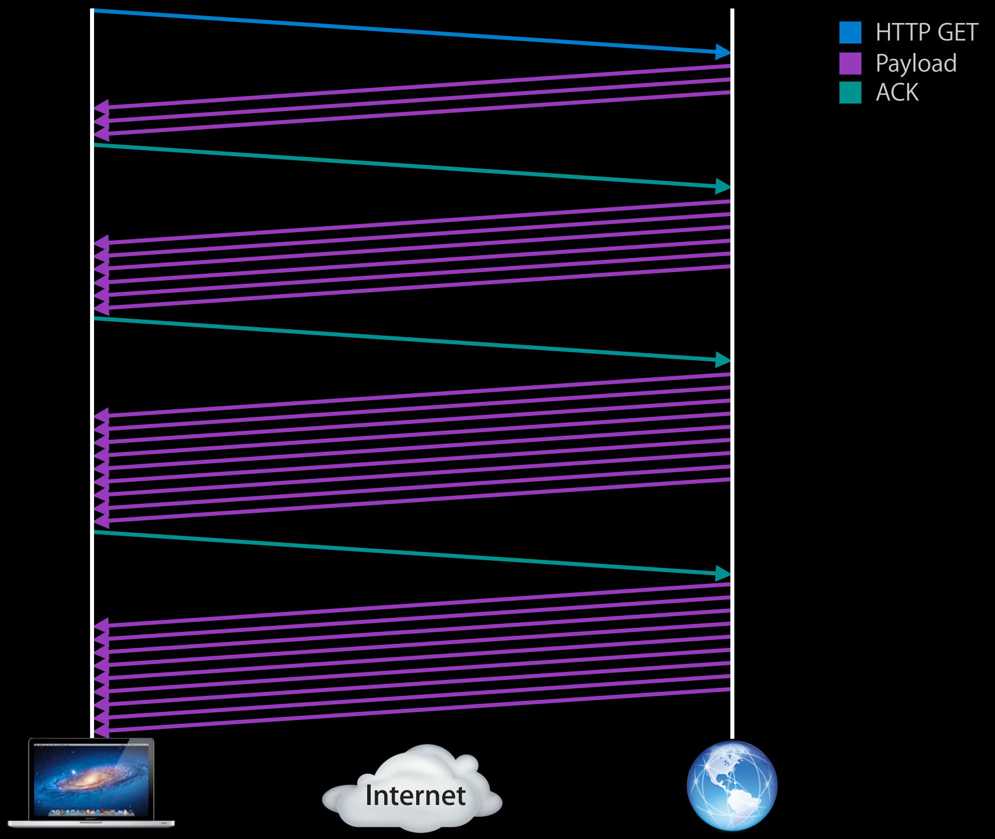
Transmission Control Protocol

Slow start



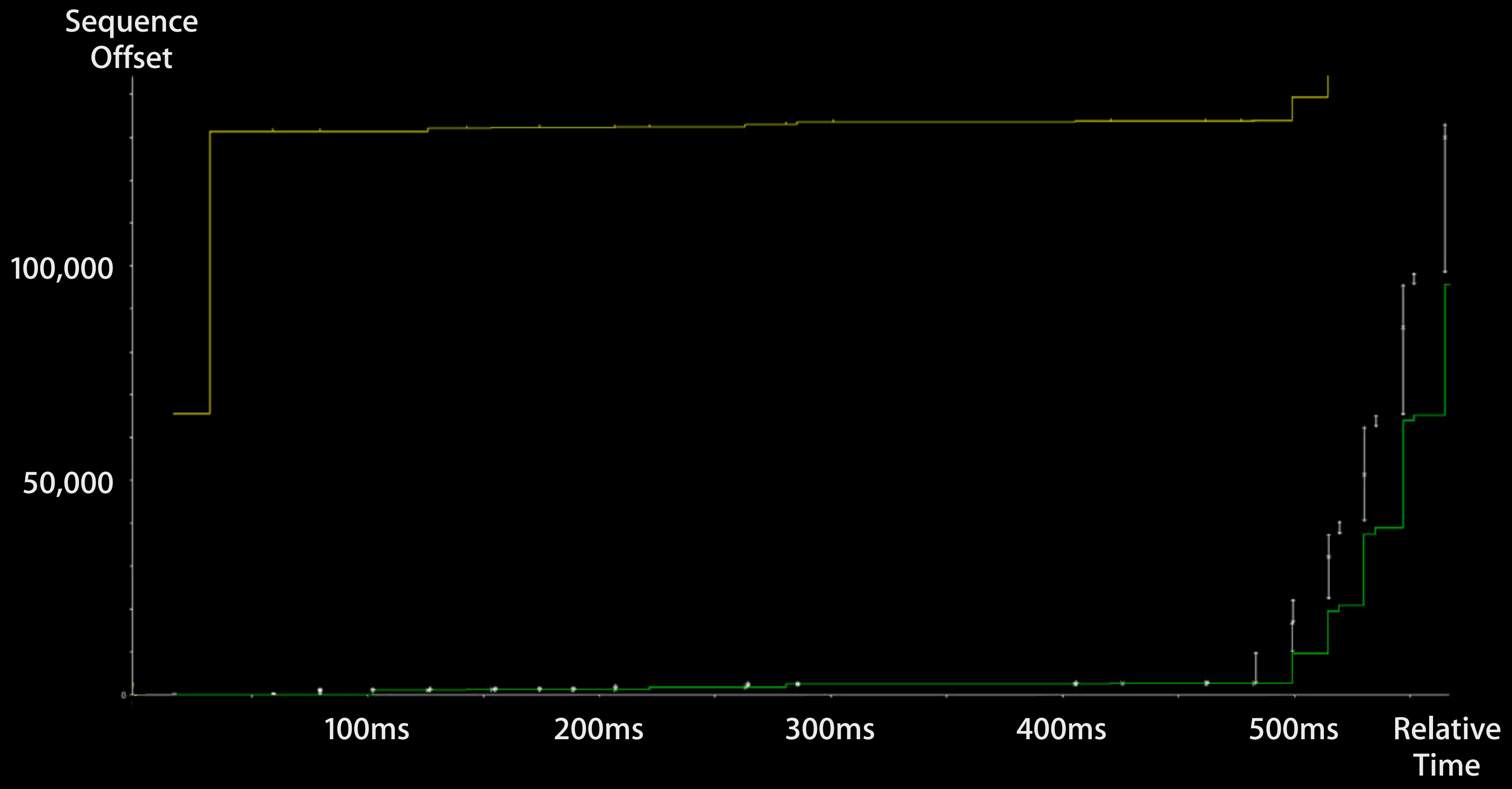
Transmission Control Protocol

Slow start



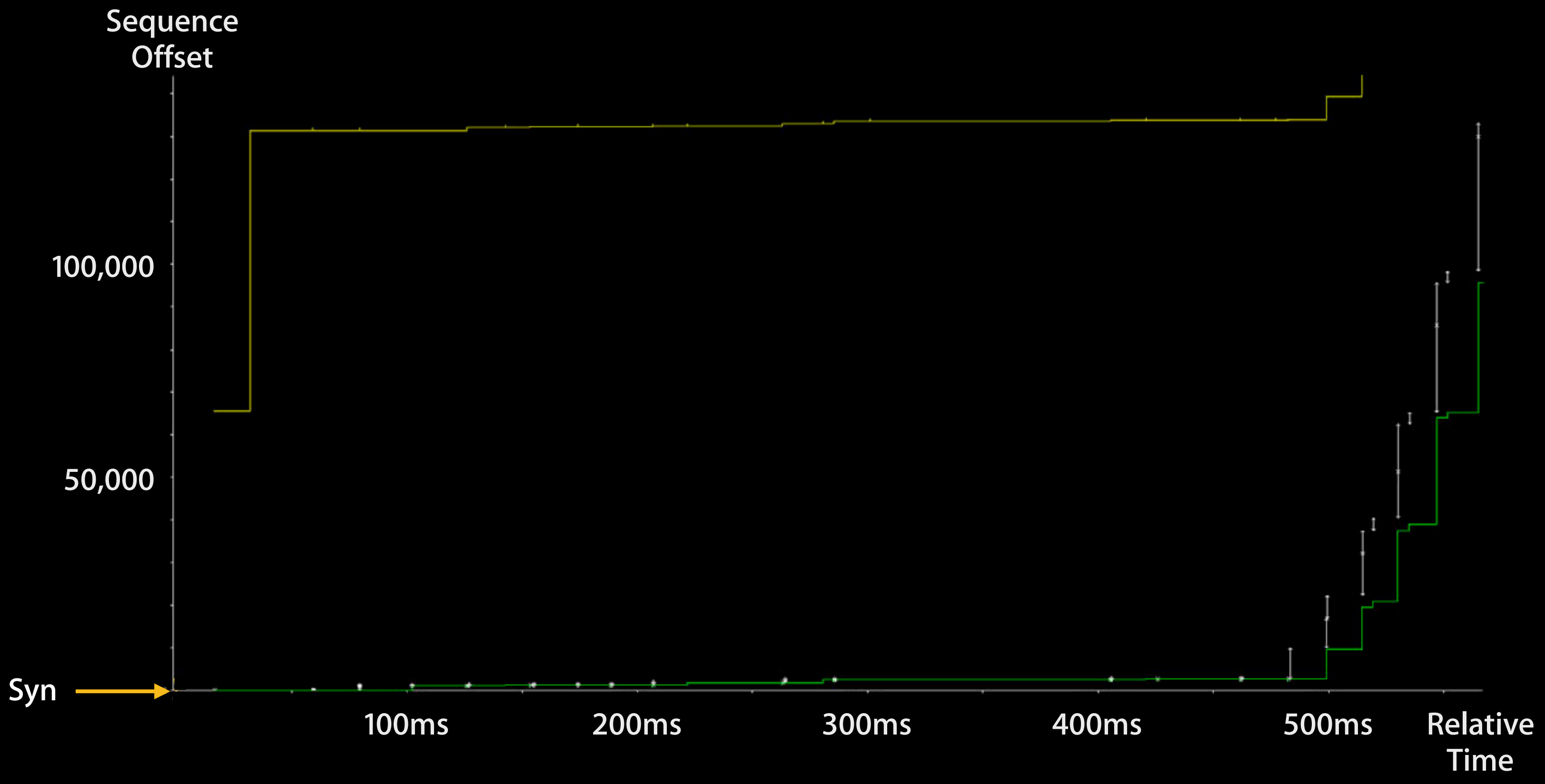
Transmission Control Protocol

tcptrace time sequence graph



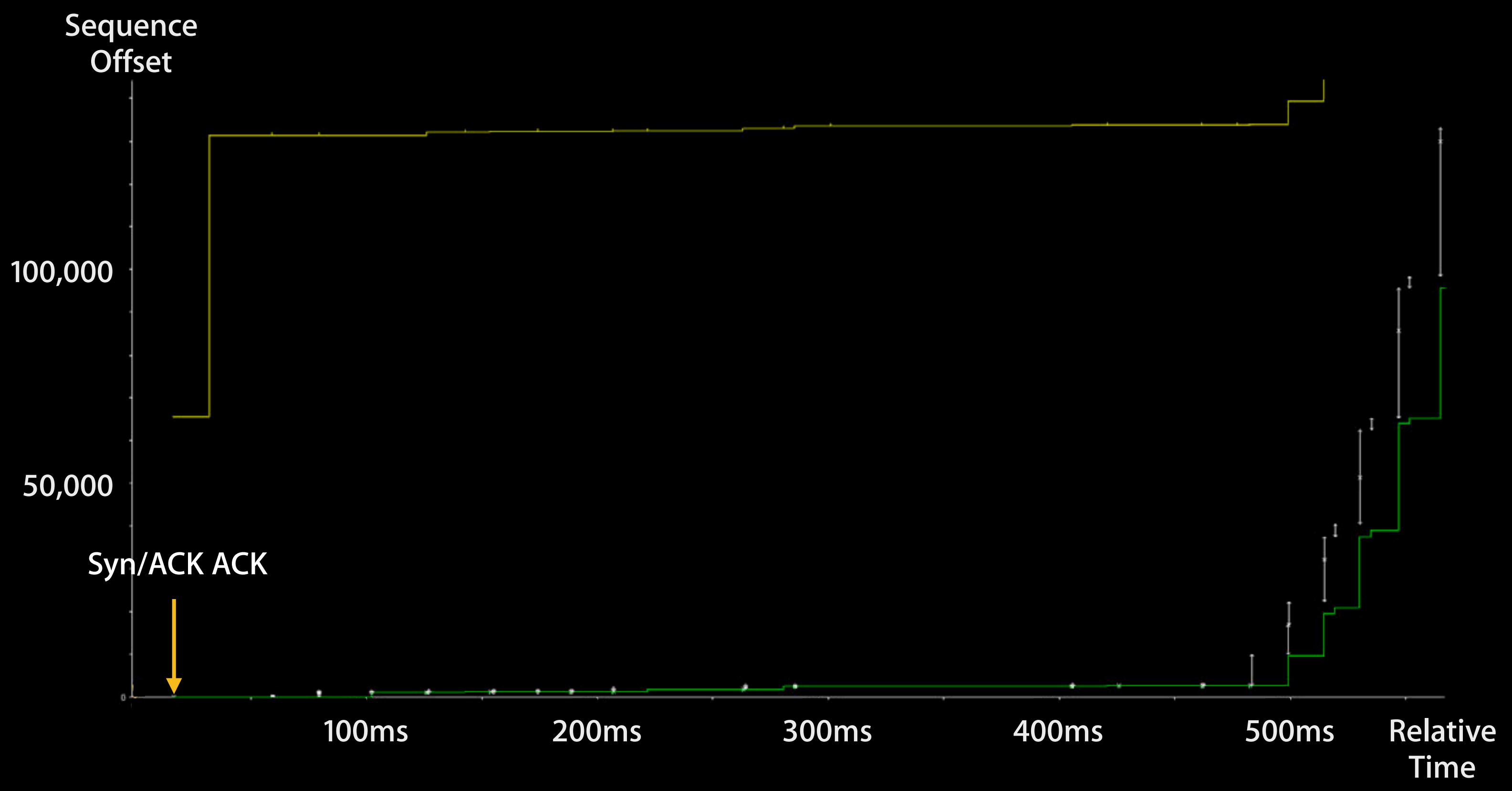
Transmission Control Protocol

tcptrace time sequence graph



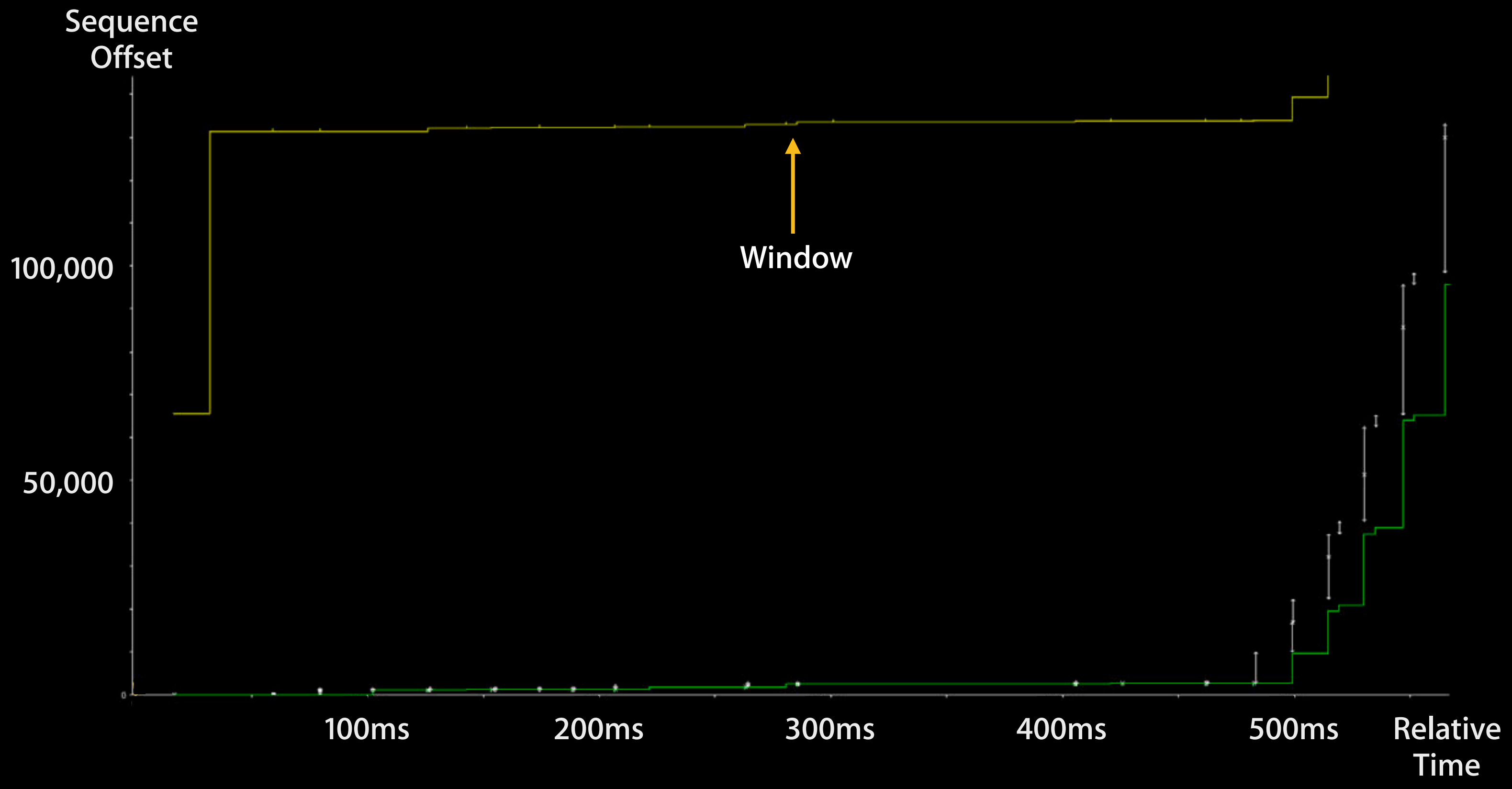
Transmission Control Protocol

tcptrace time sequence graph



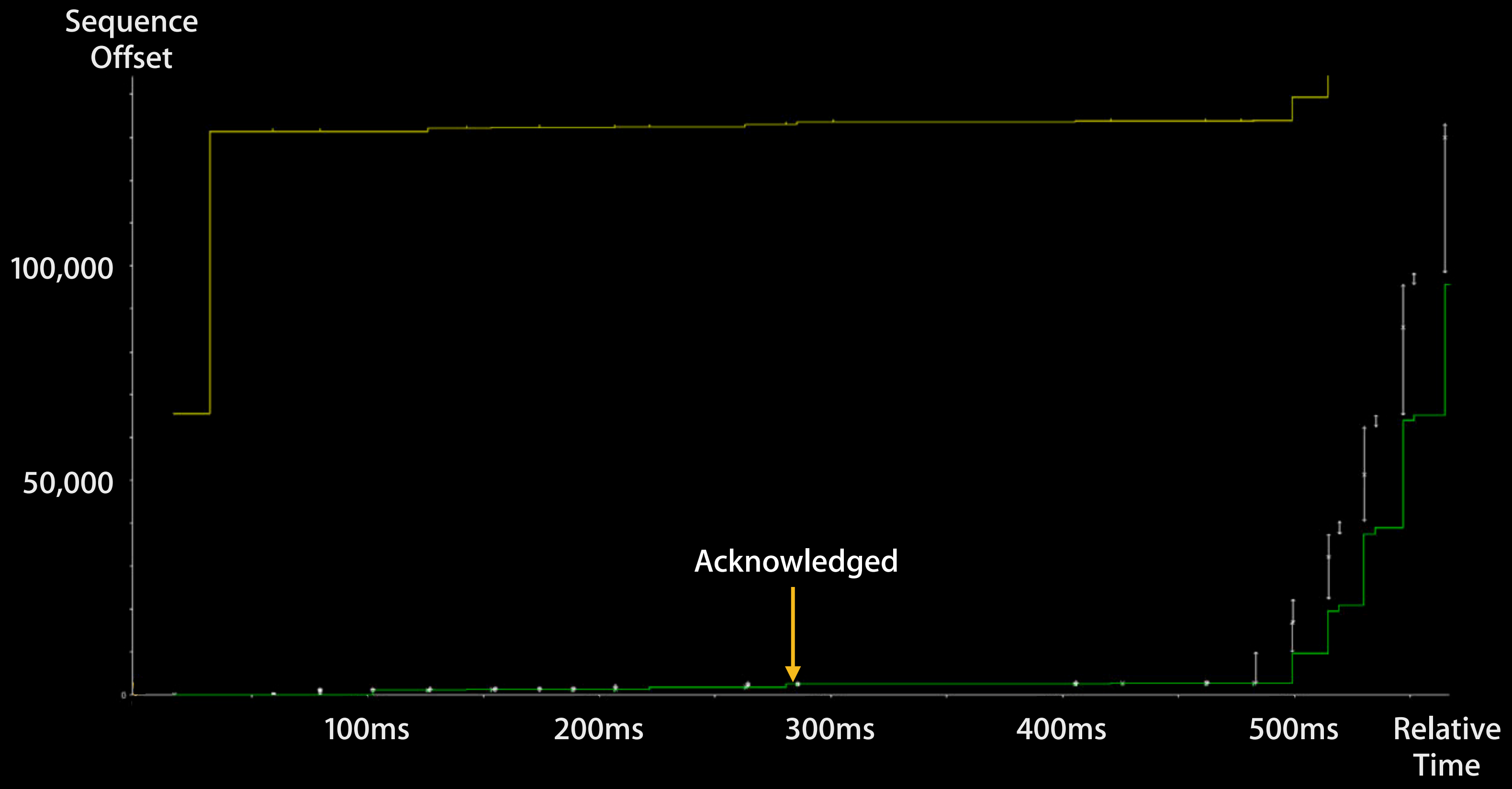
Transmission Control Protocol

tcptrace time sequence graph



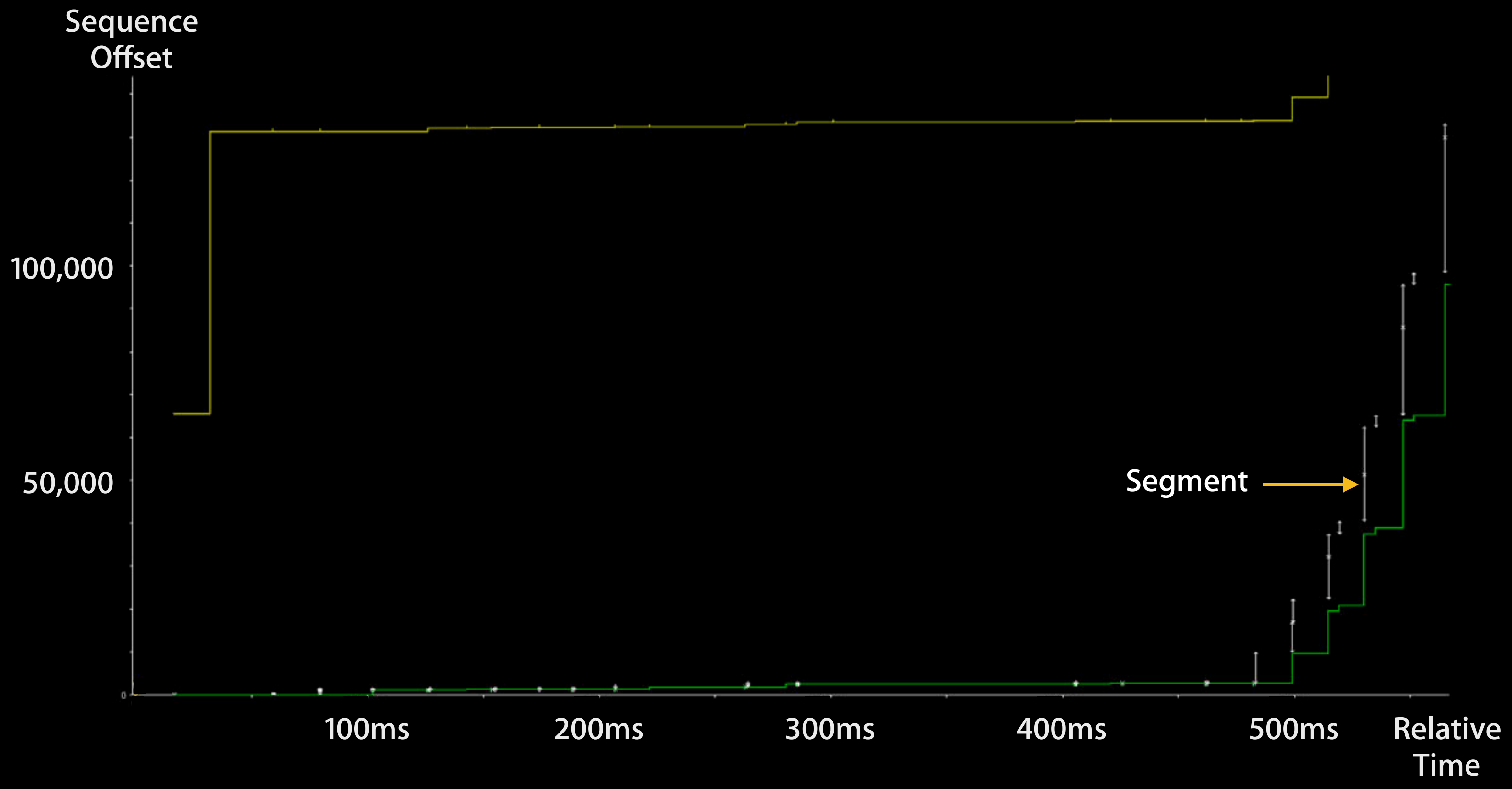
Transmission Control Protocol

tcptrace time sequence graph



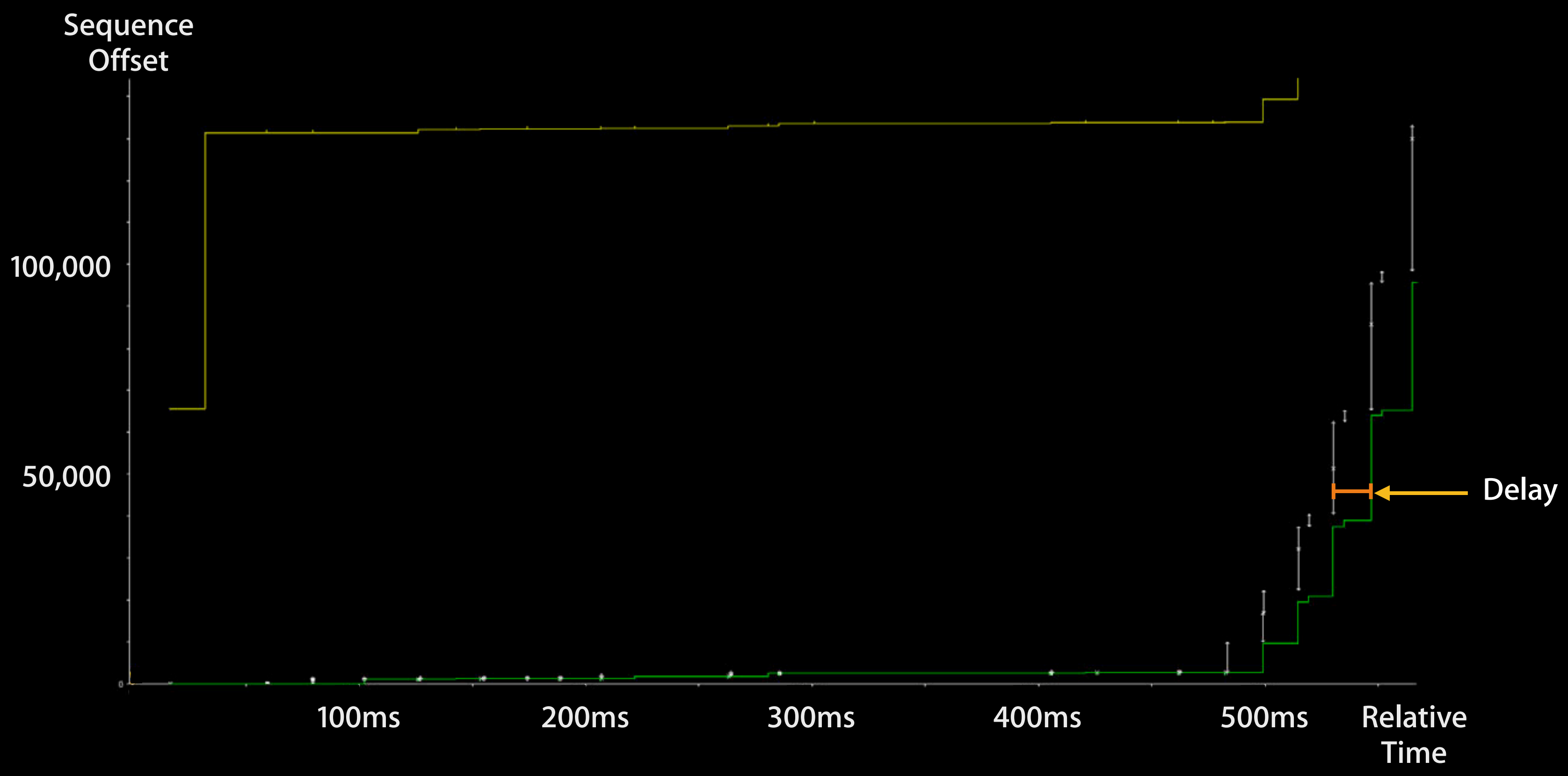
Transmission Control Protocol

tcptrace time sequence graph



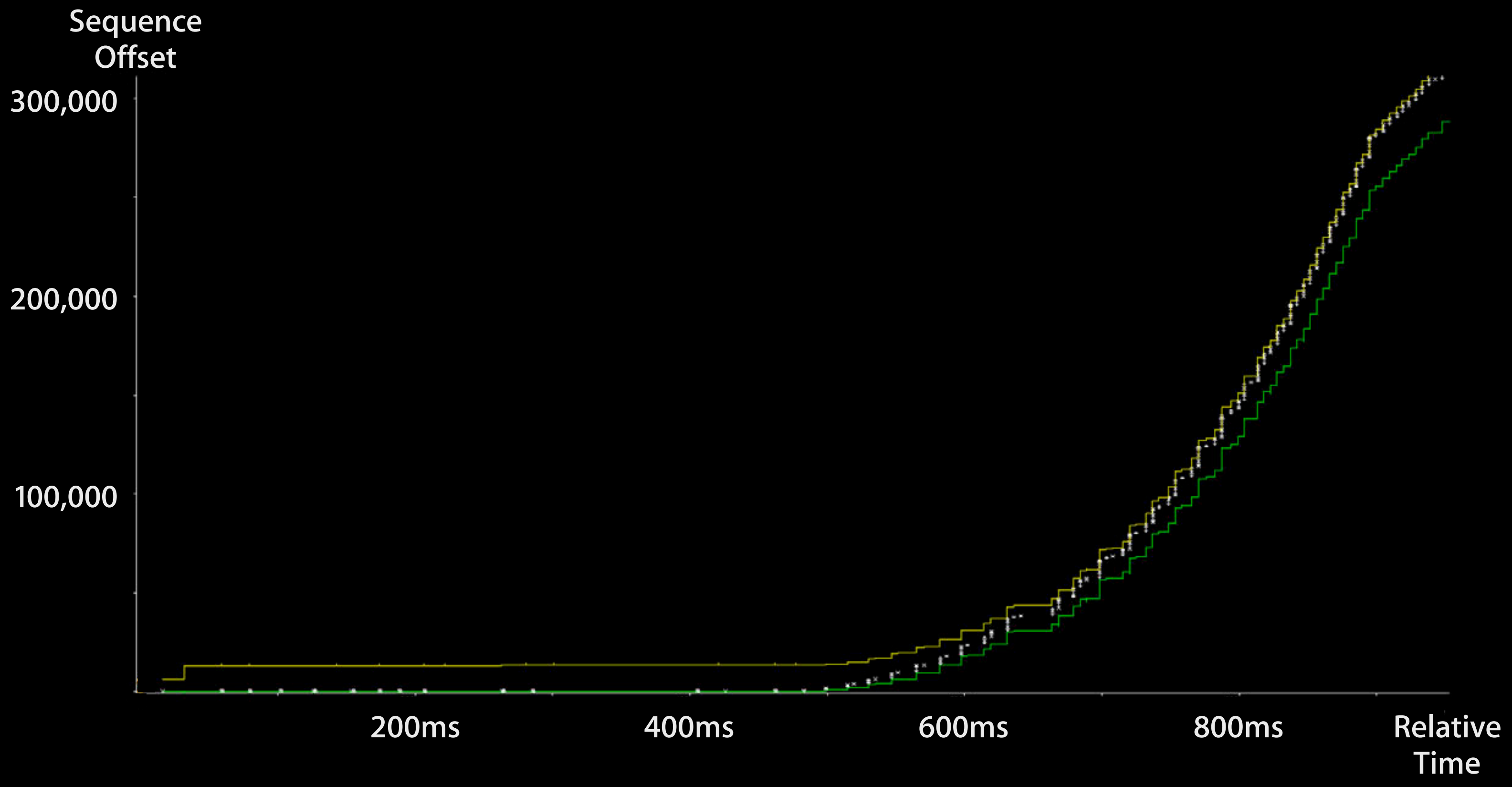
Transmission Control Protocol

tcptrace time sequence graph



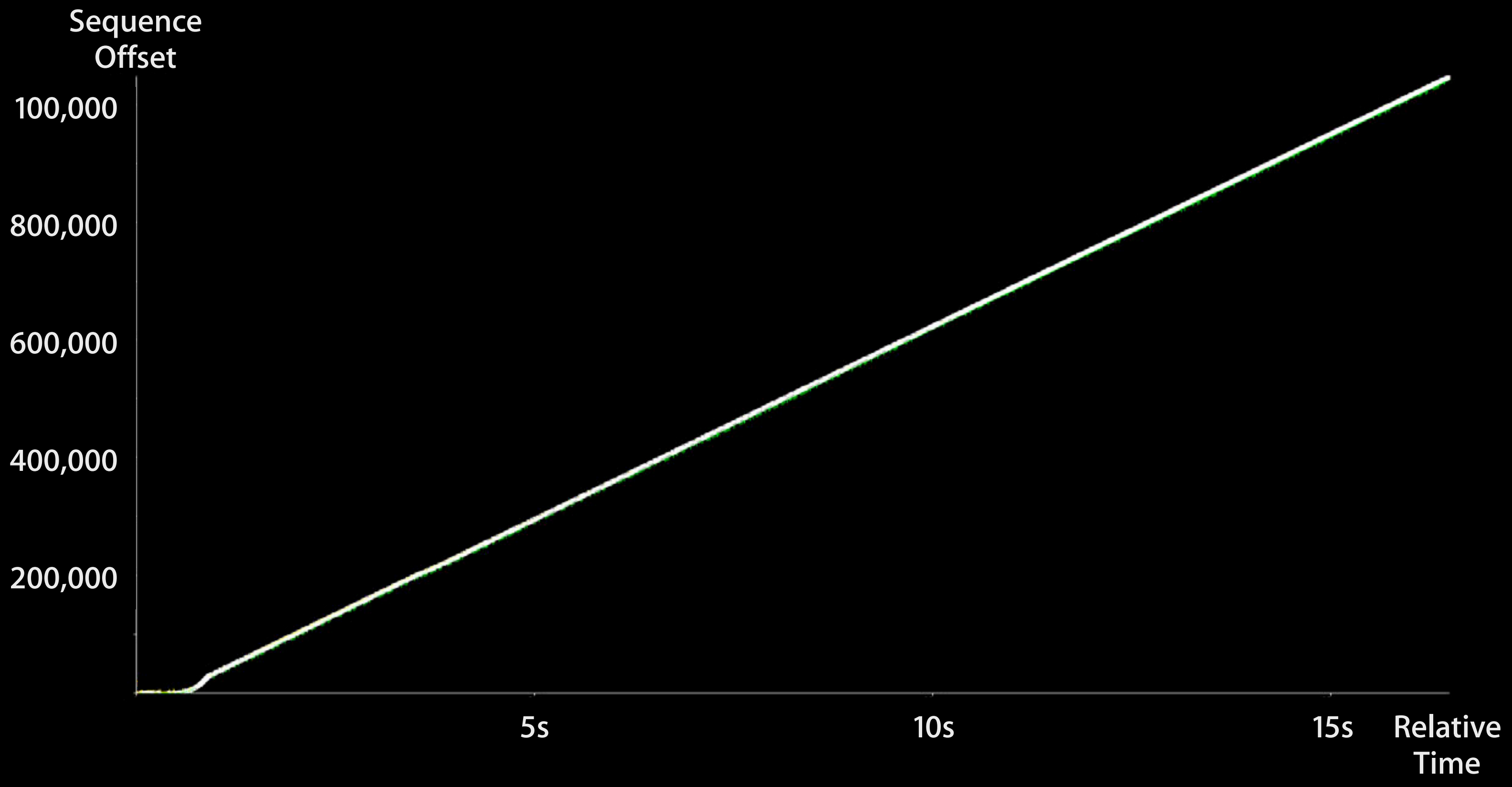
Transmission Control Protocol

Slow start



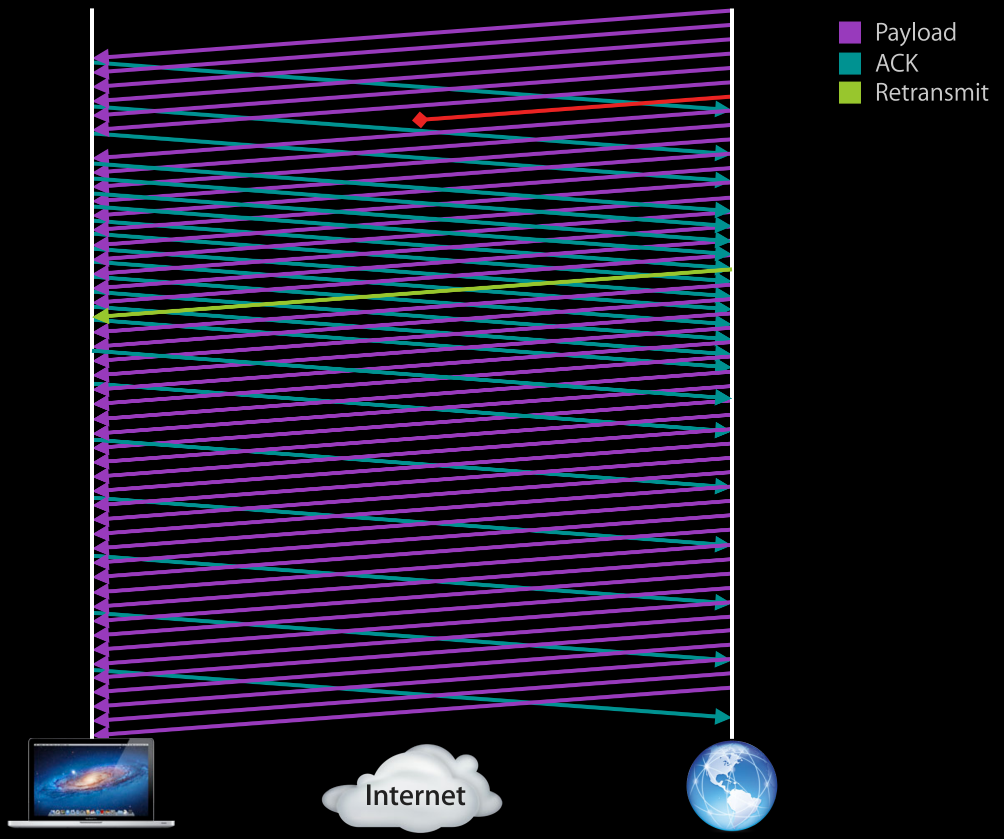
Transmission Control Protocol

Congestion avoidance



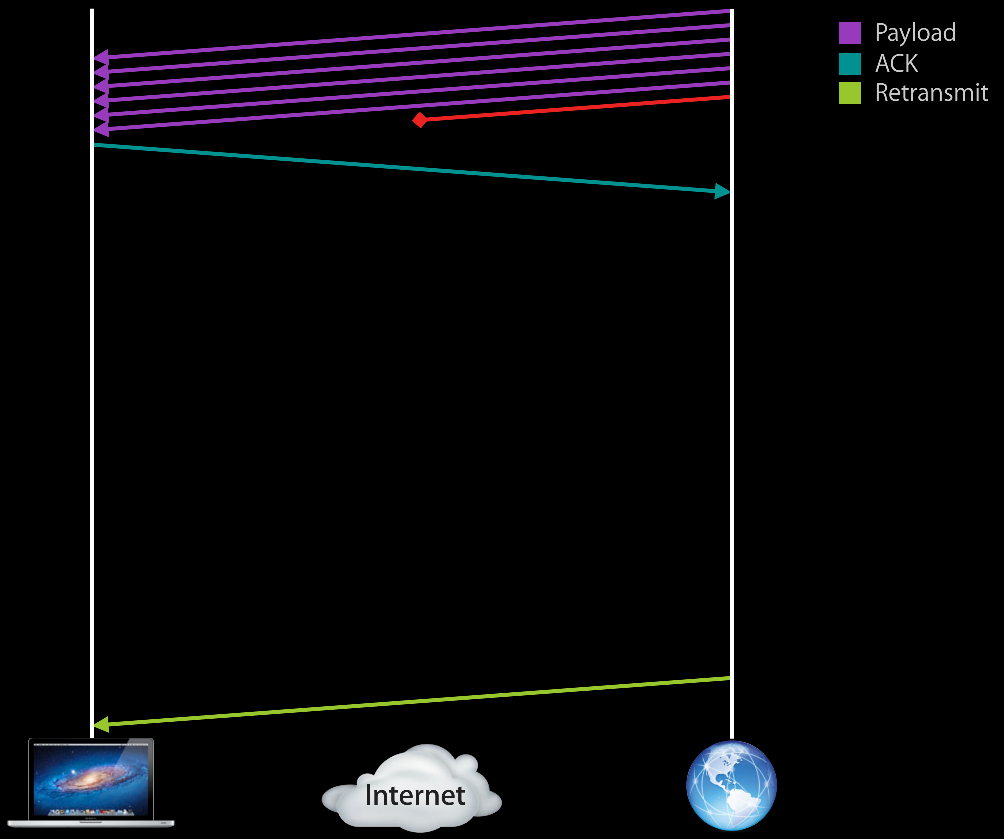
Transmission Control Protocol

Fast retransmit



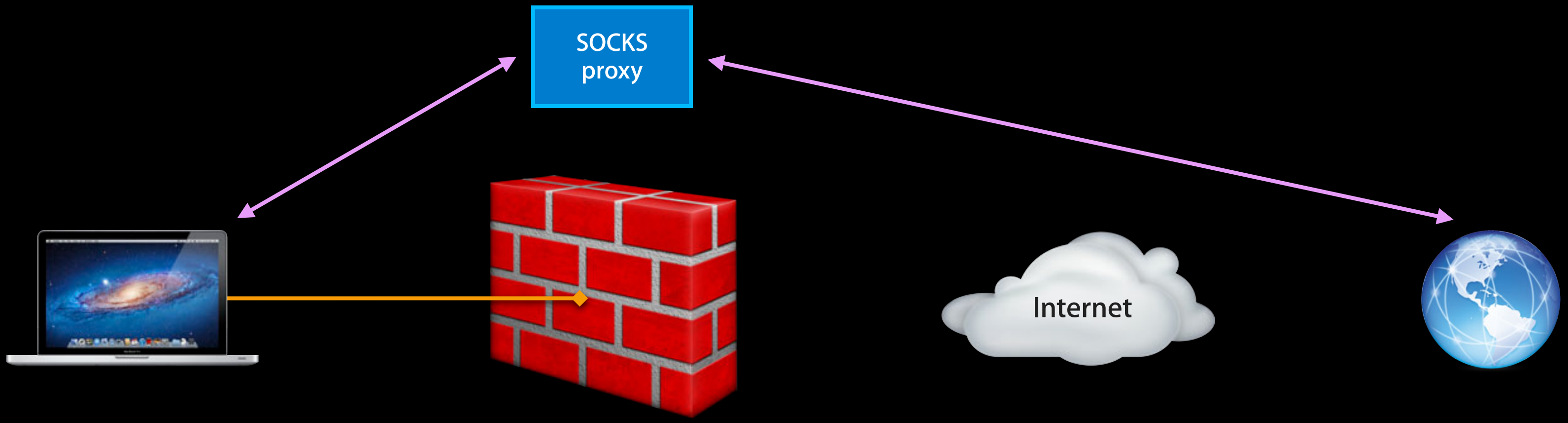
Transmission Control Protocol

Retransmit timer



Transmission Control Protocol

SOCKS proxies



Transmission Control Protocol

Best practices

- Use TCP
- Reuse TCP connections
  - New connections cost time
    - Three-way handshake
    - Slow start
    - Packet loss sensitivity
- Always keep data in flight
  - Last four packets (~5792 bytes) sensitive to loss
  - Double-buffer operations

Protocol Abstractions

Hypertext Transfer Protocol (HTTP)

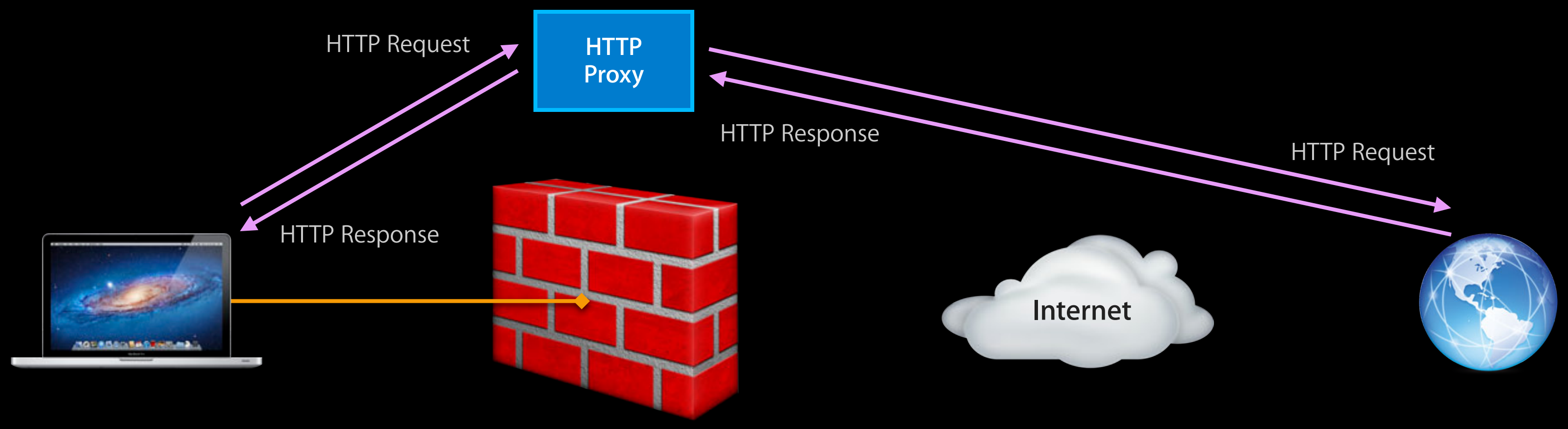
Hypertext Transfer Protocol

Services provided

- Request/response-based
- Text-based headers
    - Rich metadata
- Caching
- Proxy
- Persistent Connections
- Pipelined Requests

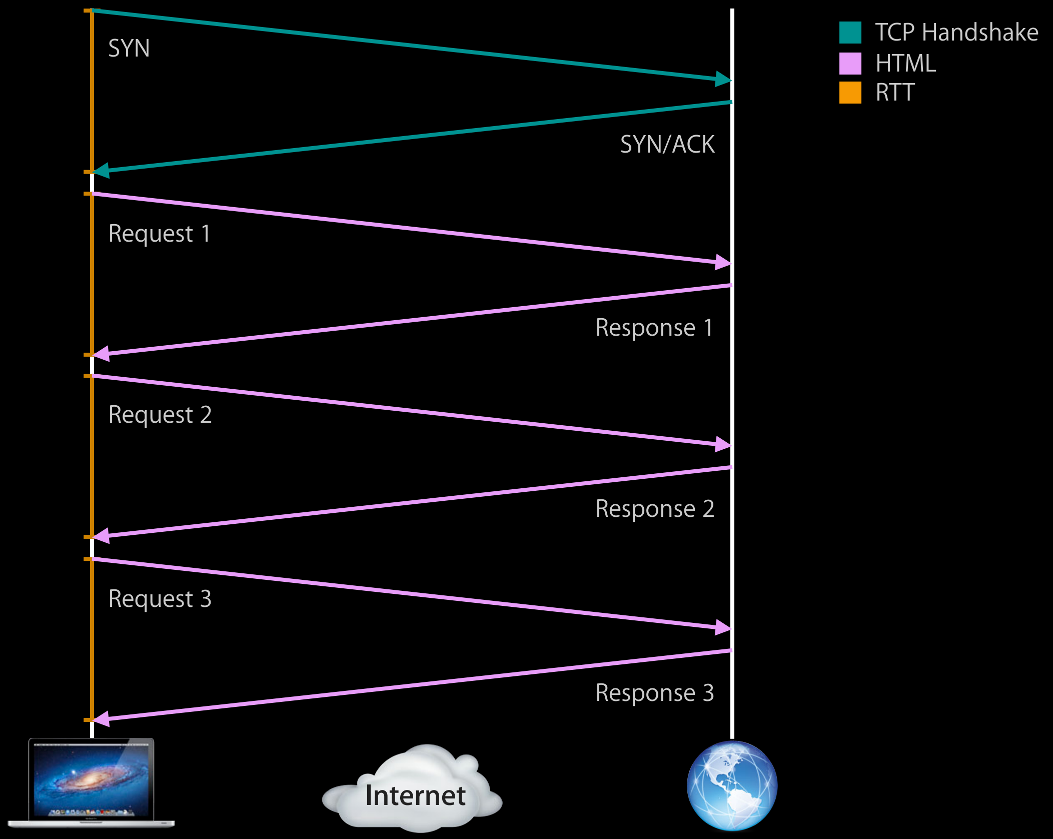
Hypertext Transfer Protocol

HTTP proxy



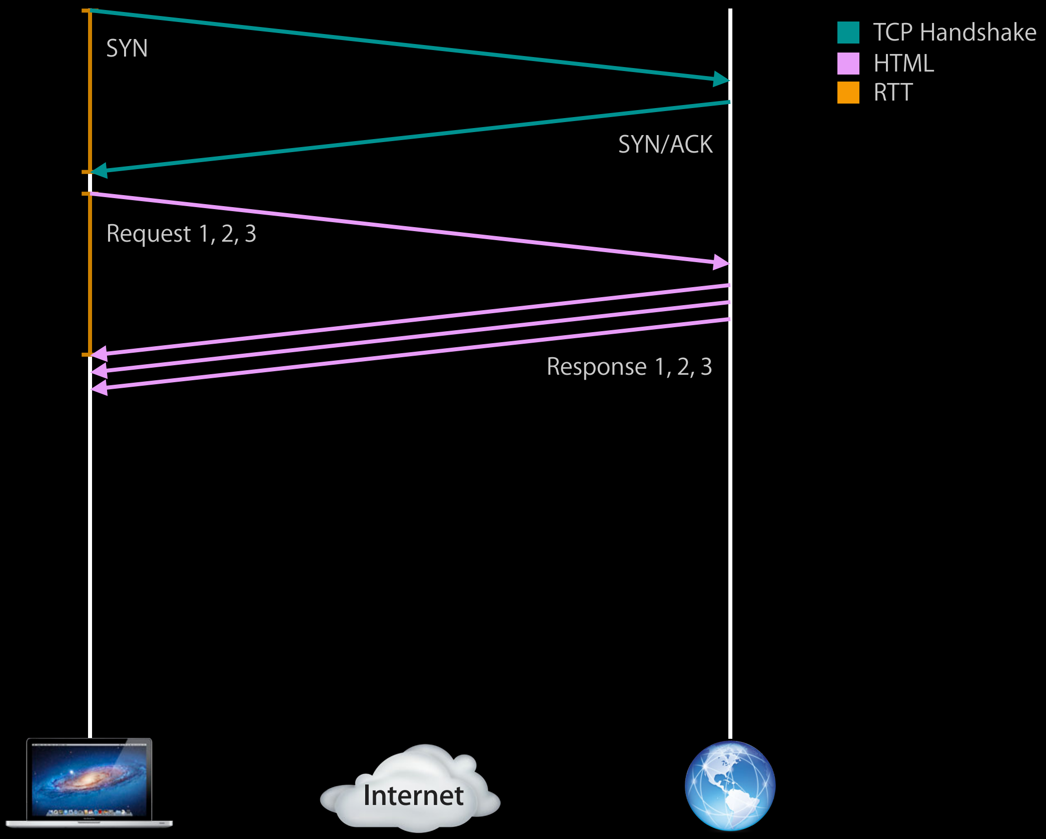
Hypertext Transfer Protocol

Persistent connection



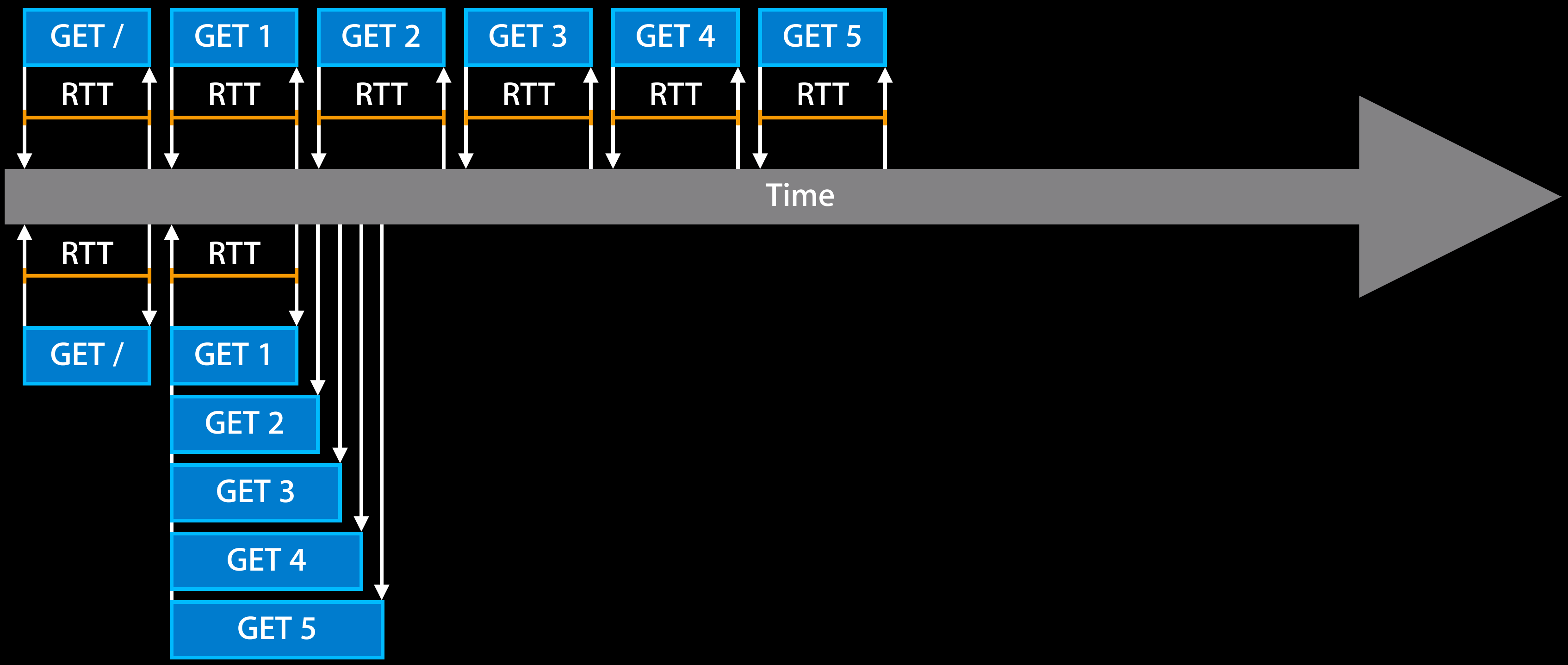
Hypertext Transfer Protocol

Pipelined connection



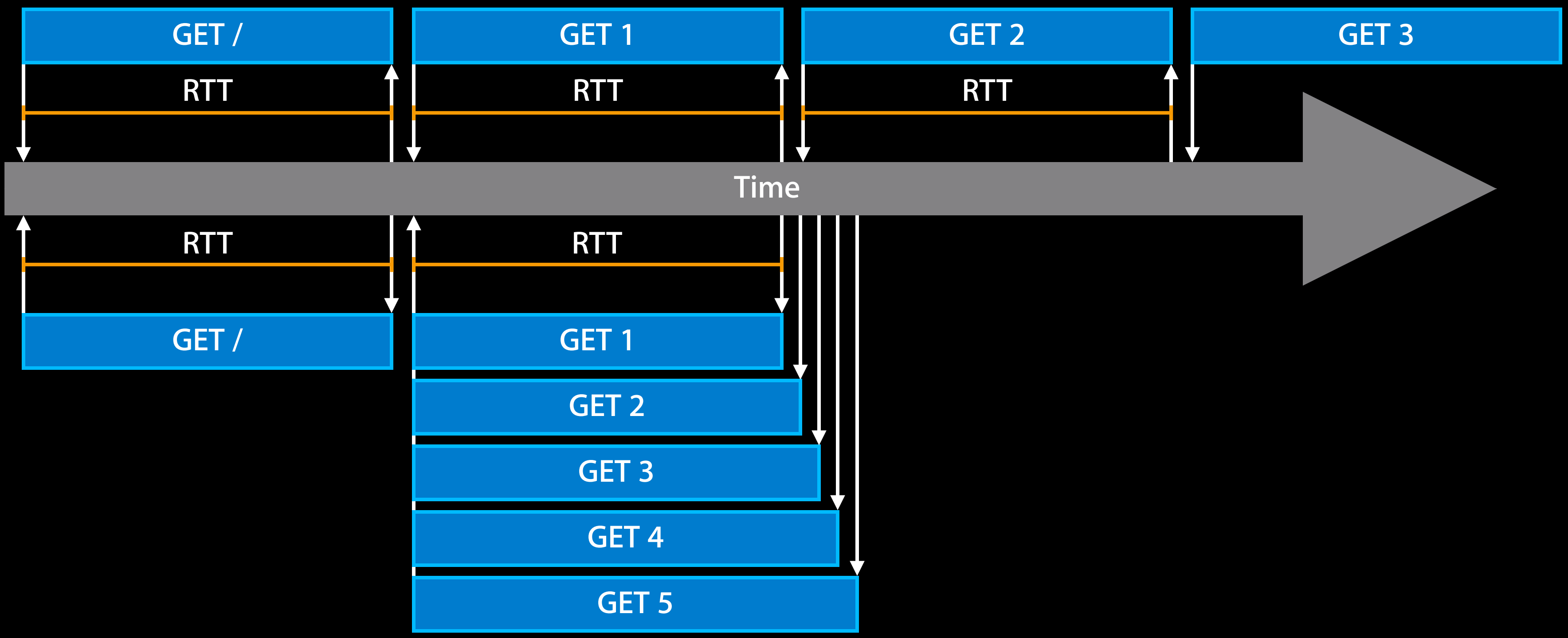
Hypertext Transfer Protocol

Pipelined connection



Hypertext Transfer Protocol

Pipelined connection



Hypertext Transfer Protocol

Best practices

- Support
  - HTTP and SOCKS proxies
  - Persistent connections
  - Pipelined requests
    - Requires server-side support

APIs

CFSocketStream

Best API for TCP

- Run loop and CF type integration
- Connect by host name
- Parallel connection attempts
    - Cellular fallback
- Cellular and VPN on-demand
- TLS and SSL
    - Server and client authentication
- SOCKS proxy
    - Fetch `CFNetworkCopySystemProxySettings`
    - Set `kCFStreamPropertySOCKSProxy`

CFSocketStream

Cellular interface

- Cellular fallback
  - SCNetworkReachability may not indicate WWAN (cellular)
  - Connection may go over cellular
- To disable cellular

  `kCFStreamPropertyNoCellular`
- To detect cellular

  `kCFStreamPropertyConnectionIsCellular`

NSInputStream and NSOutputStream

- Use
  - `CFStreamCreatePairWithSocketToHost`
  - `CFStreamCreatePairWithSocketToNetService`
- Convert CF to NS
  - CFInputStream to NSInputStream
  - CFOutputStream to NSOutputStream
  - Use `CFBridgingRelease` with ARC
- Beware NSHost (OS X only)
  - Blocking resolve on init

NSURLConnection

Best API for HTTP and HTTPS

- Asynchronous event-based API
- Features
  - Persistent connections
  - Pipelining
  - Authentication
  - Caching
  - Cookies
  - SOCKS and HTTP proxy: Automatic

NSURLConnection

Lifecycle

- Create NSURLRequest
- Send request
- Wait for response
- Wait for data
- Finish or error

NSURLConnection

- NSURLConnection ≠ TCP connection
  - Maintain pool of connections
  - Dynamically assign request to connection
  - Response may come from cache
- HTTP authentication
  - Basic, Digest, NTLM, Kerberos (OS X)
  - Automatic proxy authentication
  - `–connection: willSendRequestForAuthenticationChallenge`
- HTTP pipelining
  - `–[NSMutableURLRequest setHTTPShouldUsePipelining:]`

NSURLConnection

Caching

- Shared `[NSURLCache sharedURLCache]`
  - Simple LRU cache
  - Small in-memory (~4MB)
  - Overflows to disk (~20MB)
  - Single item 5% limit
- Tuning
  - -setMemoryCapacity
  - -setDiskCapacity
- New in iOS 6
  - On-disk cache supports https

WebKit

Best API for rendering web

- Features
  - Caching
  - Proxy support
  - Pipelining (iOS)

Timeouts

- There is no good timeout value
- RTT < 1 millisecond to > 30 seconds
- Giving up can be a disservice
  - Ordering WWDC tickets
  - Purchasing concert tickets
- Allow user to timeout
  - Attempt until user gives up
  - Retry on behalf of the user
    - On reachability changes

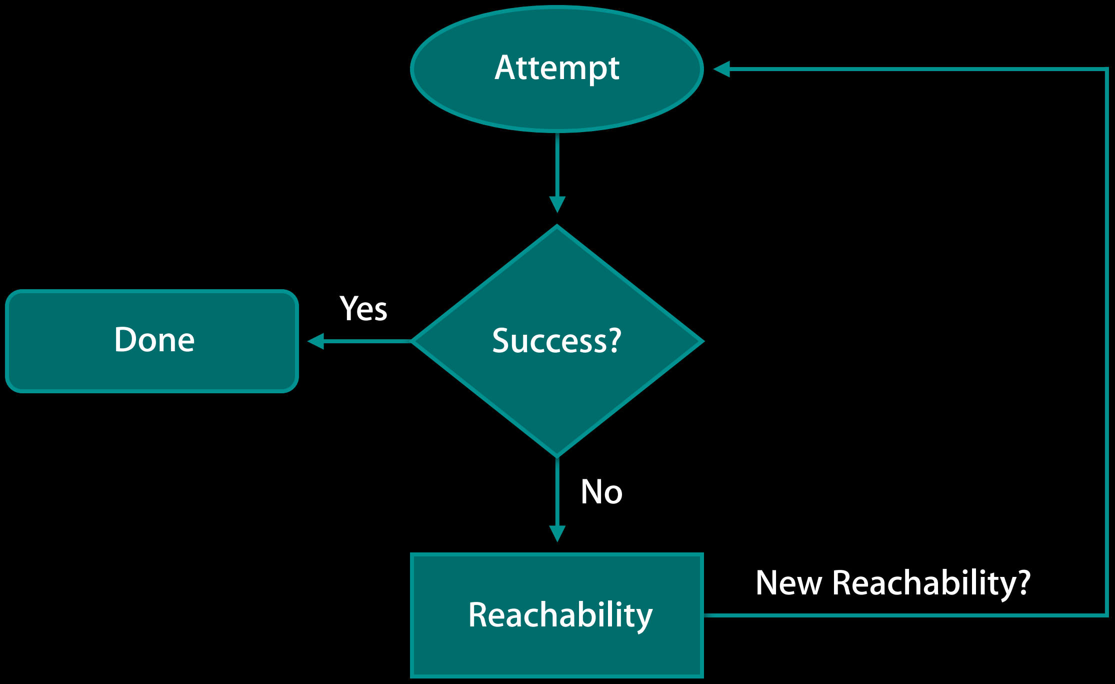
Mobility and Cost

Mobility

Challenges

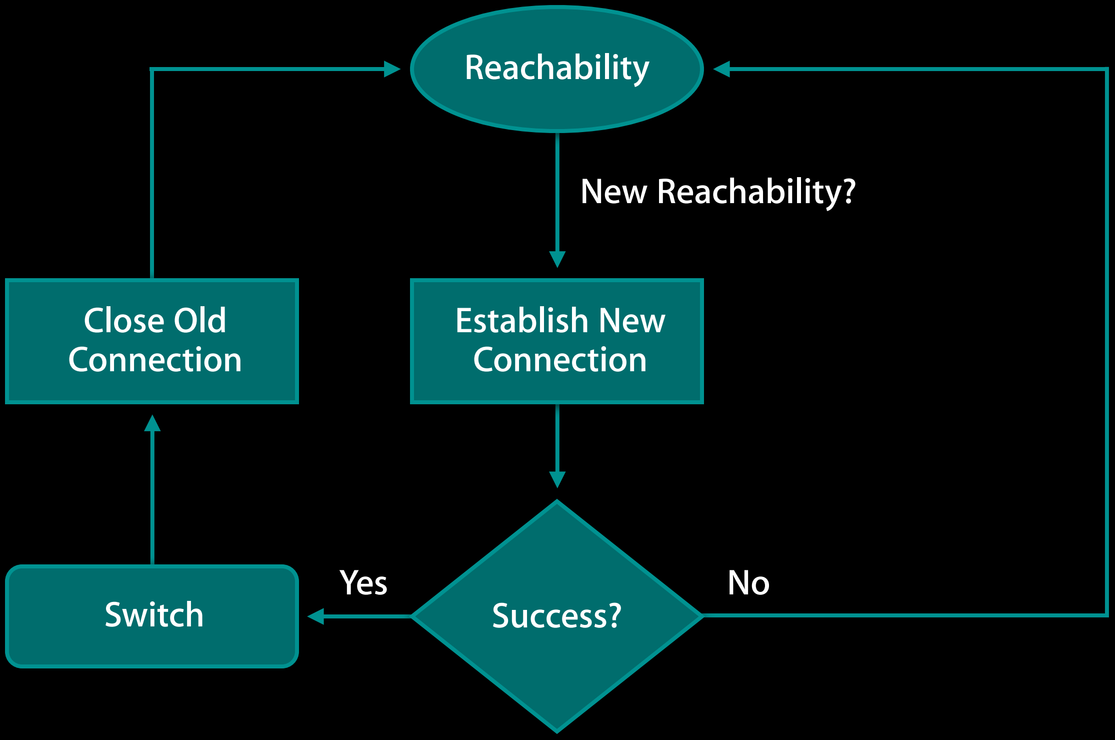
- Computers fit in pockets
- Multiple interfaces
    - Ethernet
    - Wi-Fi
    - Cellular
- Changing environment
    - Train through tunnel
    - Arriving home to Wi-Fi

Mobility
Connecting



Mobility

Connected



Cost

- Cellular
  - Power cost
  - Money cost
- Wi-Fi
  - Power cost
  - Money cost?

Cost

Solutions

- Money
  - Cache data
  - Fetch appropriately sized resources
  - Fetch only what is necessary
- Power
  - Fetch in bursts

Debugging

Debugging

- Logging
  - CFNetwork
  - libsystem_network
- Packet trace
  - tcptrace
- TLS/SSL bypass: Do not ship
- Network Link Conditioner

Debugging

CFNetwork

- CFNETWORK_DIAGNOSTICS environment variable
  - CFNETWORK_DIAGNOSTICS=1
    - Internal CFNetwork events and state
  - CFNETWORK_DIAGNOSTICS=2
    - Adds make/reuse TCP connection decisions
  - CFNETWORK_DIAGNOSTICS=3
    - Adds TLS/SSL decrypted content logging
    - Use with caution
    - CFNETWORK_IO_LOG_FILE=<path>
- Output to file and syslog

Debugging

libsystem_network

- Debug
    - Connection problems
    - CFSocketStream and above
- Enable logging

```
sudo defaults write /Library/Preferences/com.apple.networkd
libnetcore_log_level -int 7
```

- Disable logging

```
sudo defaults delete /Library/Preferences/com.apple.networkd
libnetcore_log_level
```

- Display logging

```
syslog -w
```

Debugging

Packet trace

- tcpdump
    - OS X
        - New, show pid: -k
    - iOS
        - Start: `rvictl -s <UDID>`
        - `tcpdump -i rvi0`
        - Stop: `rvictl -x <UDID>`
    - Write to file: `-w <file>`

Debugging

tcptrace

- Download and build
  - http://tcptrace.org.
- Capture packets with tcpdump -w
- Create .xpl files using tcptrace
  - TCP
    - Run tcptrace -G -n -zxy packets.pcap
  - HTTP
    - Run tcptrace -xHTTP -n -zxy packets.pcap
- Open .xpl in jPlot or xplot

Debugging

Network Link Conditioner

- OS X
  - Hardware IO Tools for Xcode
- iOS (new for iOS 6)
  - Enable device for development
  - Settings->Developer->Network Link Conditioner
- Switch to slow/lossy network
  - tcpdump
  - Test (early and often)
  - tcptrace
  - Verify optimal utilization

More Information

Apple WWDC 2010
Networking Apps for iPhone OS, Part 1 and 2
https://developer.apple.com/videos/wwwdc/2010

Related Sessions

Simplify Networking with Bonjour	Nob Hill Tuesday 4:30PM

Labs

Networking Lab	Core OS Lab A Wednesday 9:00AM
Networking Lab	Core OS Lab A Thursday 9:00AM

Summary

- Use TCP
  - Reuse TCP connections
  - Multiple concurrent requests on single connection
    - Fast retransmits
    - Hide latency
  - Support SOCKS proxies
- Use HTTP
  - Use pipelining
  - Support SOCKS and HTTP proxies

WWDC2012