System Frameworks #WWDC15

Your App and Next Generation Networks

Session 719

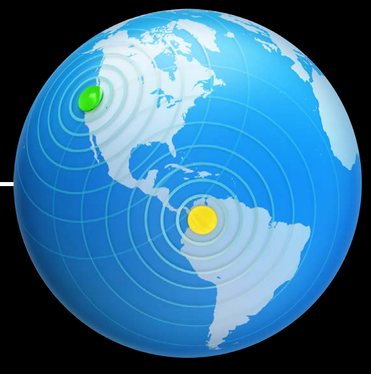
Prabhakar Lakhera Core OS Networking Engineer
Stuart Cheshire DEST

Part One
Transitioning to IPv6-Only Networks

Part Two
Reducing Delays in Networking

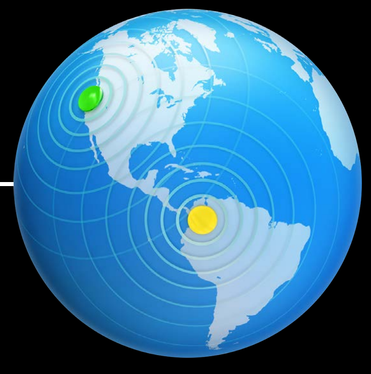
Transitioning to IPv6-Only Networks

Cellular Data Network



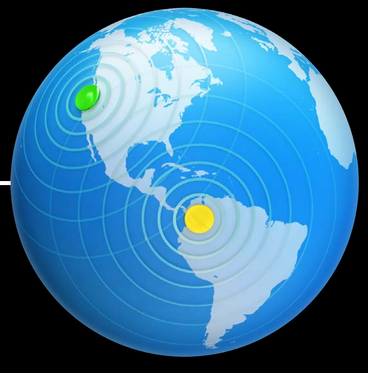
IPv4 Server

Cellular Data Network

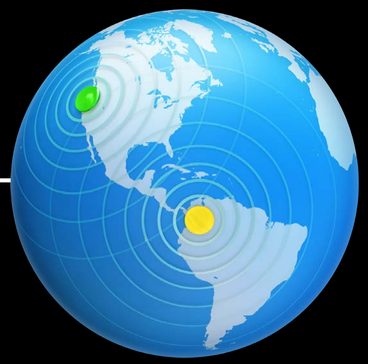


IPv4 Server

Cellular Data Network

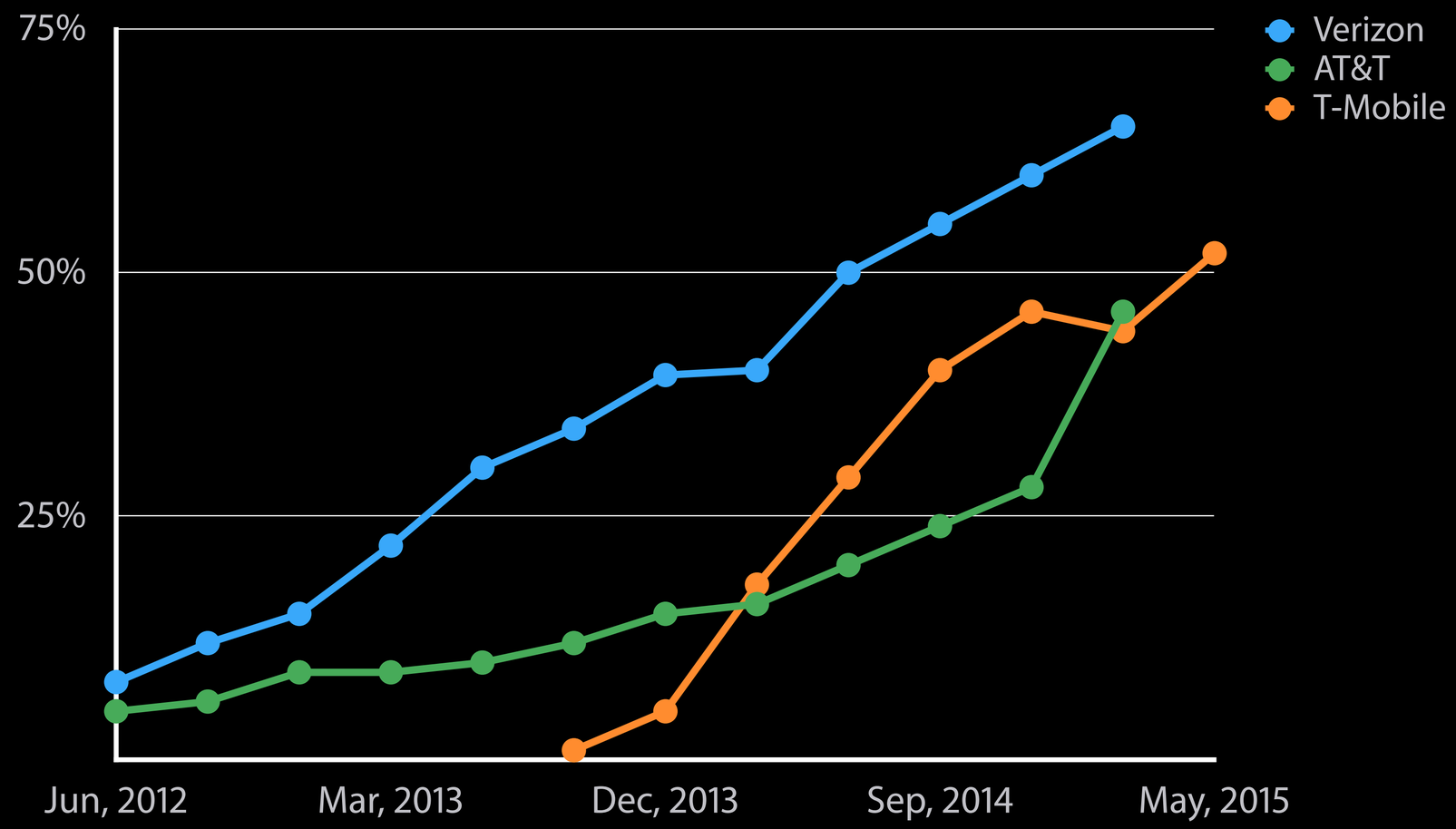


IPv4 Server

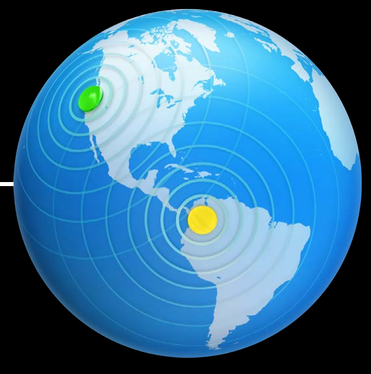


IPv6 Server

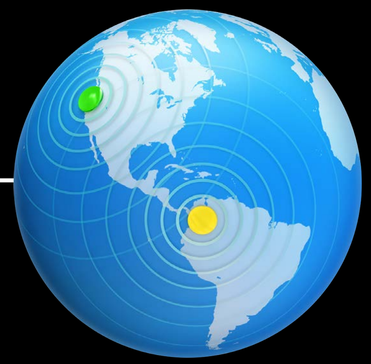
Cellular Data Network



Cellular Data Network

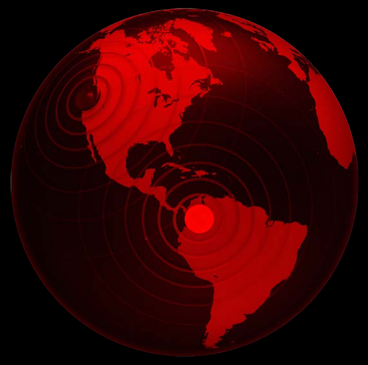


IPv4 Server



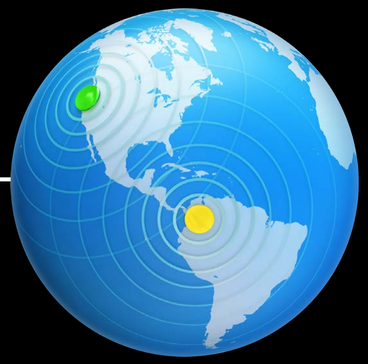
IPv6 Server

Cellular Data Network



IPv4 Server

IPv6 Access Connectivity

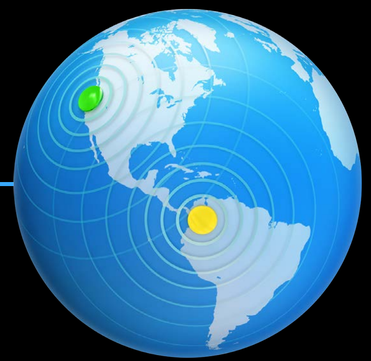


IPv6 Server

Cellular Data Network

DNS64
NAT64

IPv6 Access
Connectivity



IPv4 Server

IPv6 Server

Cellular Data Network

DNS64 synthesizes IPv6 address for IPv4 server

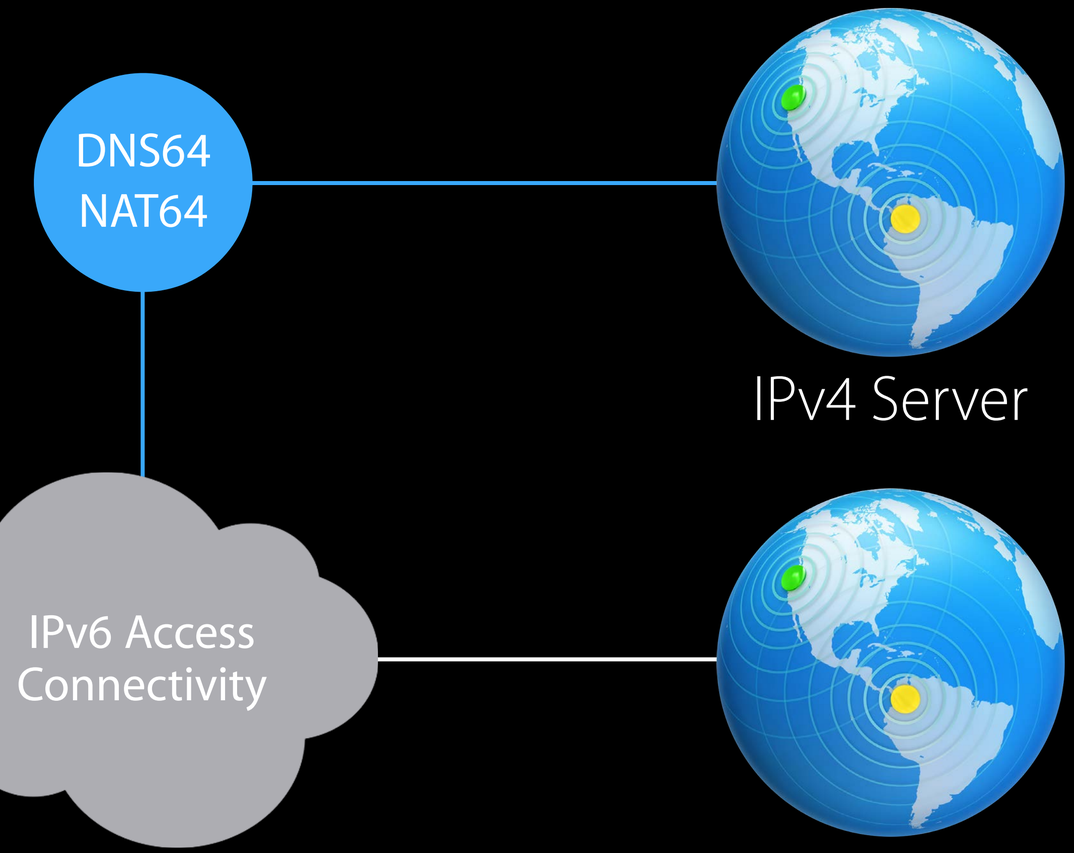
IPv4 Server

IPv6 Server

Cellular Data Network

DNS64 synthesizes IPv6 address for IPv4 server

NAT64 performs IPv6 to IPv4 address translation



Your App Has To Be IPv6 Ready

It will be an app submission requirement later this year!

System Preferences Edit View Window Help

Mon 9:41 AM

System Preferences

Search

General

Desktop & Screen Saver

Dock

Mission Control

Language & Region

Security & Privacy

Spotlight

Notifications

CDs & DVDs

Displays

Energy Saver

Keyboard

Mouse

Trackpad

Printers & Scanners

Sound

iCloud

Internet Accounts

Extensions

Network

Bluetooth

Sharing

Users & Groups

Parental Controls

App Store

Dictation & Speech

Date & Time

Startup Disk

Time Machine

Accessibility

System Font

Step 1
Option Click
Sharing

System Preferences Edit View Window Help

Mon 9:41 AM

Sharing

Search

Computer Name: WWDC

Computers on your local network can access your computer at:
WWDC.local

Edit...

On	Service
☐	DVD or CD Sharing
☐	Screen Sharing
☐	File Sharing
☐	Printer Sharing
☑	Remote Login
☐	Remote Management
☐	Remote Apple Events
☐	Internet Sharing
☐	Bluetooth Sharing

Internet Sharing: Off

Internet Sharing allows other computers to share your connection to the Internet. Computers connected to AC power won't sleep while Internet Sharing is turned on.

Share your connection from: Ethernet

To computers using:

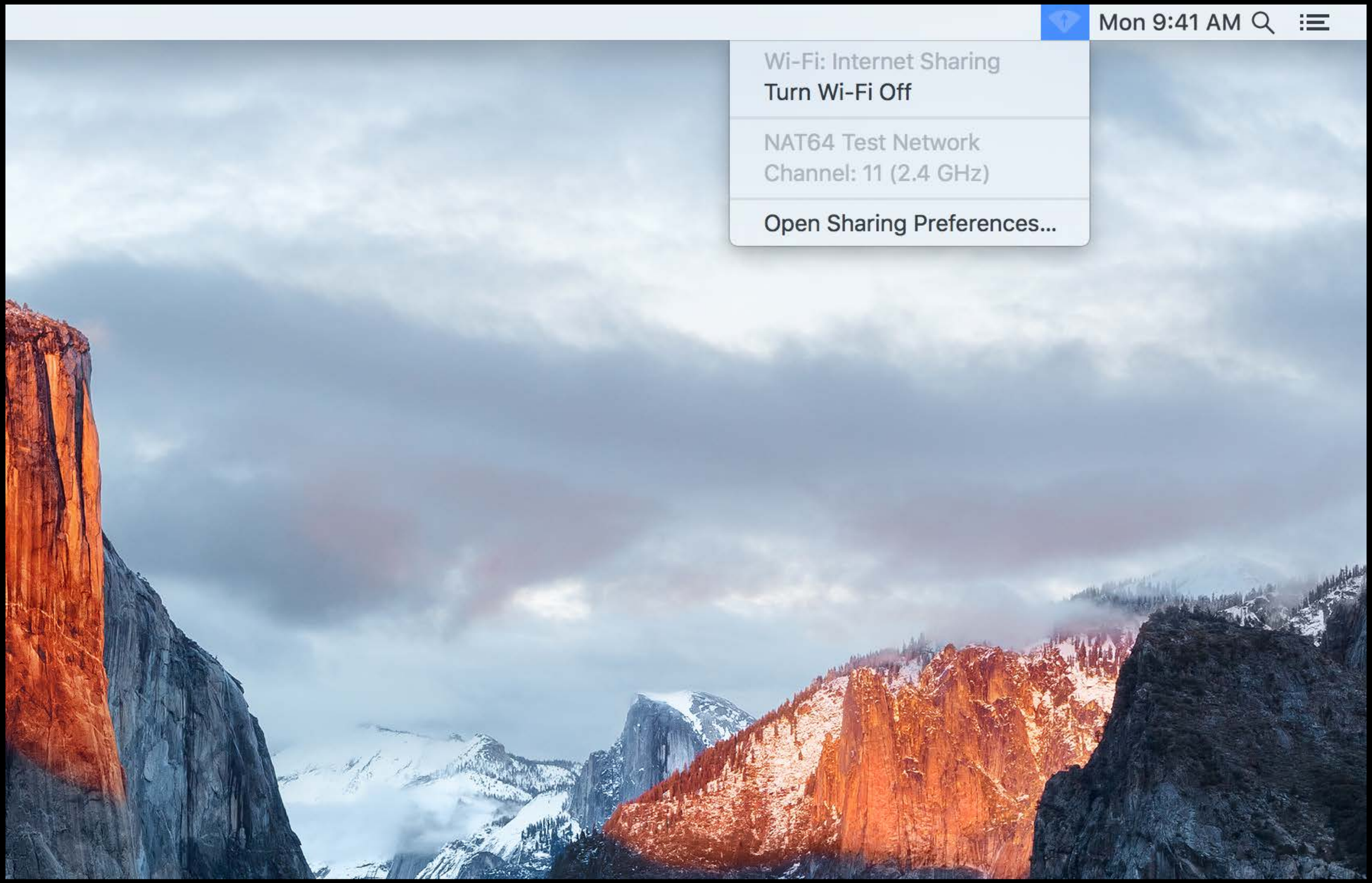
On	Ports
☑	Wi-Fi
☐	Ethernet
☐	Bluetooth PAN
☐	FireWire

☑ Create NAT64 Network

Wi-Fi Options...

Step 2
Option Click
Internet Sharing

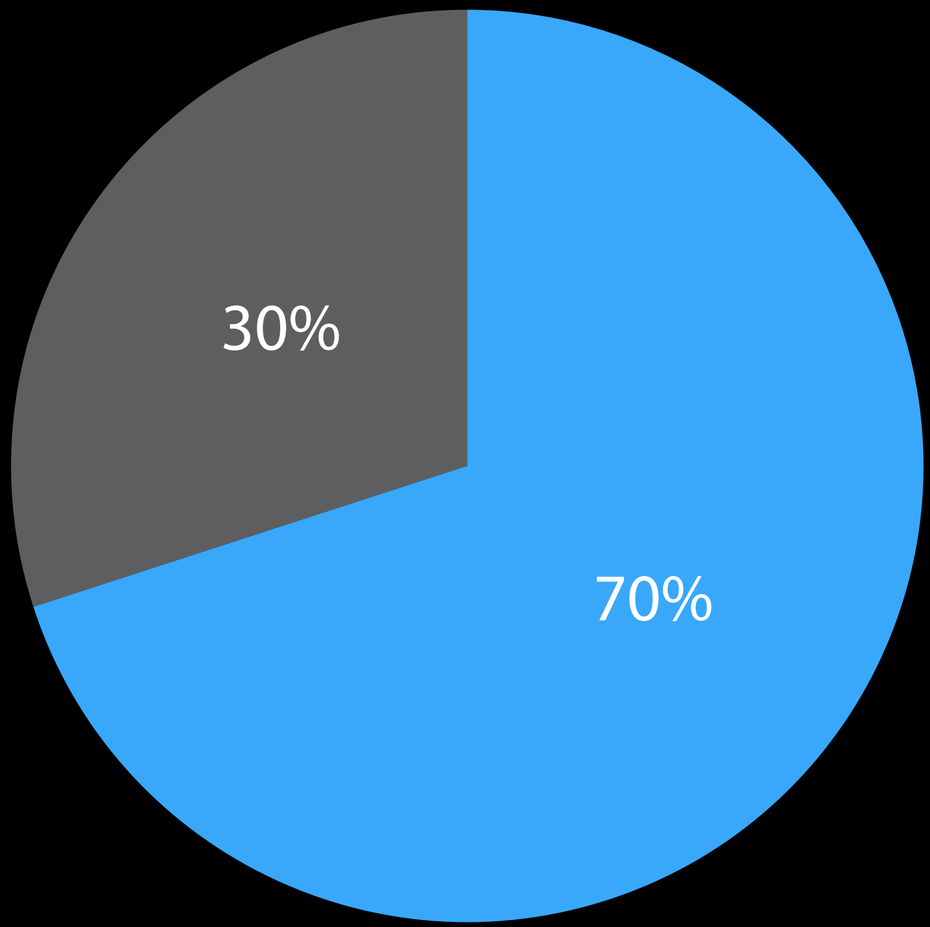
Step 3
Turn on NAT64


Mon 9:41 AM
Wi-Fi: Internet Sharing
Turn Wi-Fi Off
NAT64 Test Network
Channel: 11 (2.4 GHz)
Open Sharing Preferences...

NAT64 + DNS64 Internet Sharing

Make NAT64 Testing Part of Your Regular Development Process

Top 100 Free iOS Applications



* Results for Top 100 Free iOS Applications that need Networking

What Breaks?

IPv4-only code

IPv4-only storage objects: **uint32_t, in_addr, sockaddr_in**

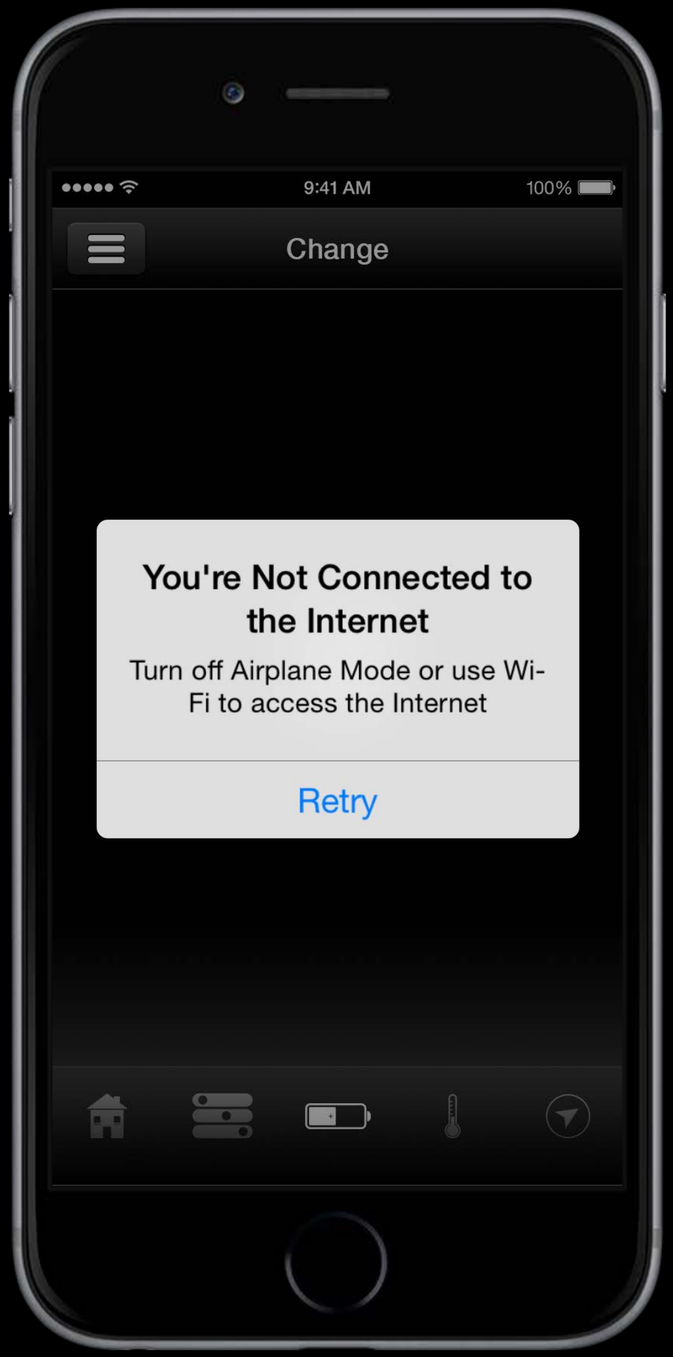
IPv4-only APIs: **inet_aton, gethostbyname**

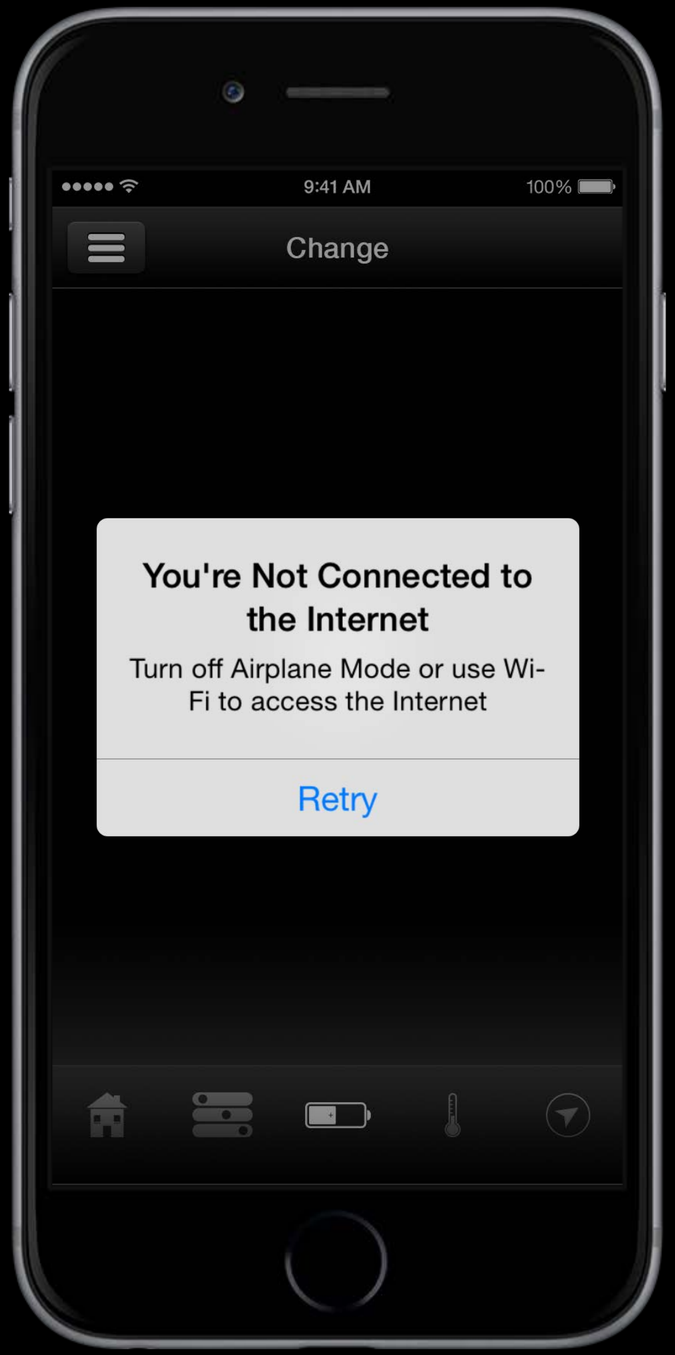
IPv4-only usage of an API: **gethostbyname2(hostname, AF_INET);**

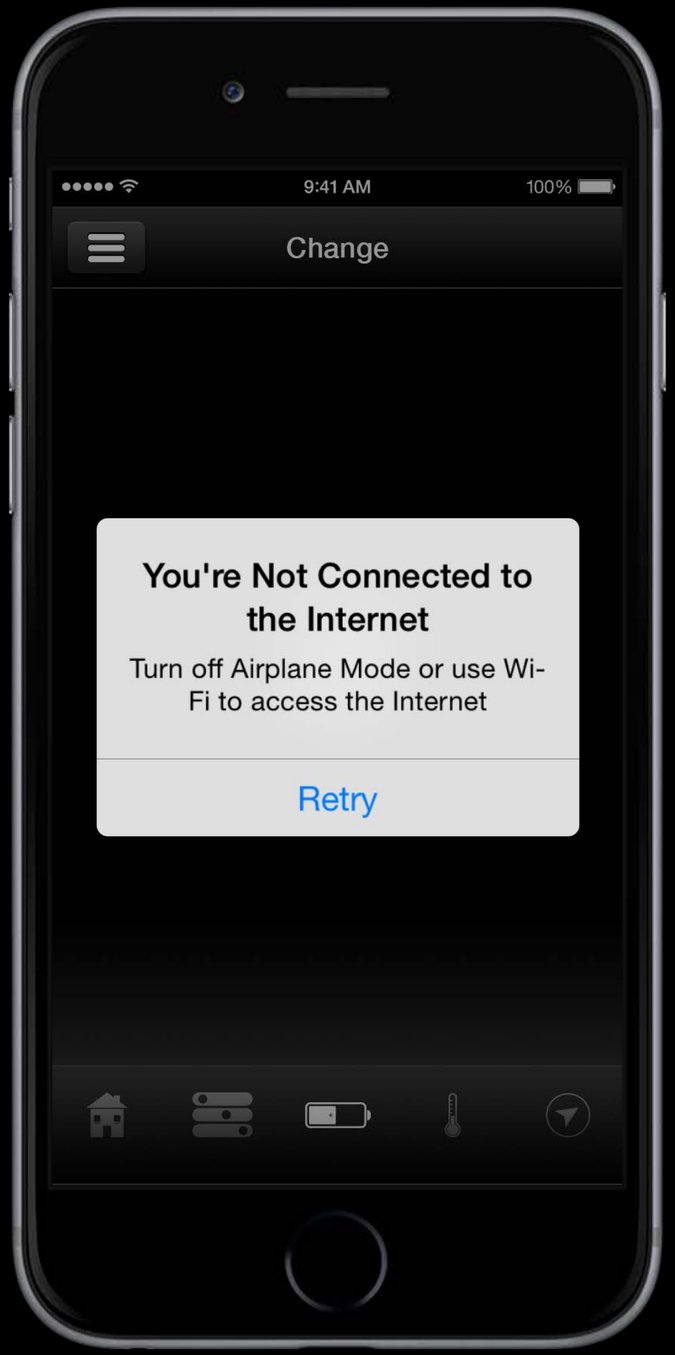
Pre-flight checks before connecting

- Checking if device has an IPv4 address
- Checking for reachability to 0.0.0.0


9:41 AM
100%
Change


9:41 AM
100%
Change
You're Not Connected to the Internet
Turn off Airplane Mode or use Wi-Fi to access the Internet
Retry


9:41 AM
100%
Change
You're Not Connected to the Internet
Turn off Airplane Mode or use Wi-Fi to access the Internet
Retry


9:41 AM
100%
Change
You're Not Connected to the Internet
Turn off Airplane Mode or use Wi-Fi to access the Internet
Retry

What Works?

Address-family agnostic code

Connect without pre-flight

- If connection succeeds, great
- If connection fails, handle that gracefully

Use higher-layer networking frameworks

- NSURLSession and CFNetwork-layer APIs

RFC 4038 “Application Aspects of IPv6 Transition”

Connect-by-name APIs

What Works?

IPv4 address literals, in NAT64 + DNS64 networks

New for OS X 10.11 and iOS 9

Use higher-layer networking frameworks

- NSURLSession and CFNetwork-layer APIs

Client supplies IPv4 address Literal

- OS synthesizes IPv6 address

Reducing Delays in Networking

Delay Reduction

Delay Reduction

Reliable Network Fallback

Explicit Congestion Notification

TCP_NOTSENT_LOWAT

TCP Fast Open

Delay Reduction

Reliable Network Fallback

Reduce Connection Setup Stalls

Explicit Congestion Notification

TCP_NOTSENT_LOWAT

TCP Fast Open

Reliable Network Fallback

Fringe of Wi-Fi

TCP connection not succeeding

OS initiates parallel connection over mobile data

First to complete wins—like RFC 6555 (Happy Eyeballs)

Reliable Network Fallback

Fully automatic

No more bill shock

Use NSURLSession and CFNetwork-layer APIs

For best user experience:

- Better Route Notification

Networking with NSURLSession	Pacific Heights	Thursday 9:00AM

Delay Reduction

Reliable Network Fallback

Reduce Connection Setup Stalls

Explicit Congestion Notification

Reduce Network Delays

TCP_NOTSENT_LOWAT

TCP Fast Open

Test: 10Mb/s Downstream

256kB FIFO queue with Tail Drop

VS.

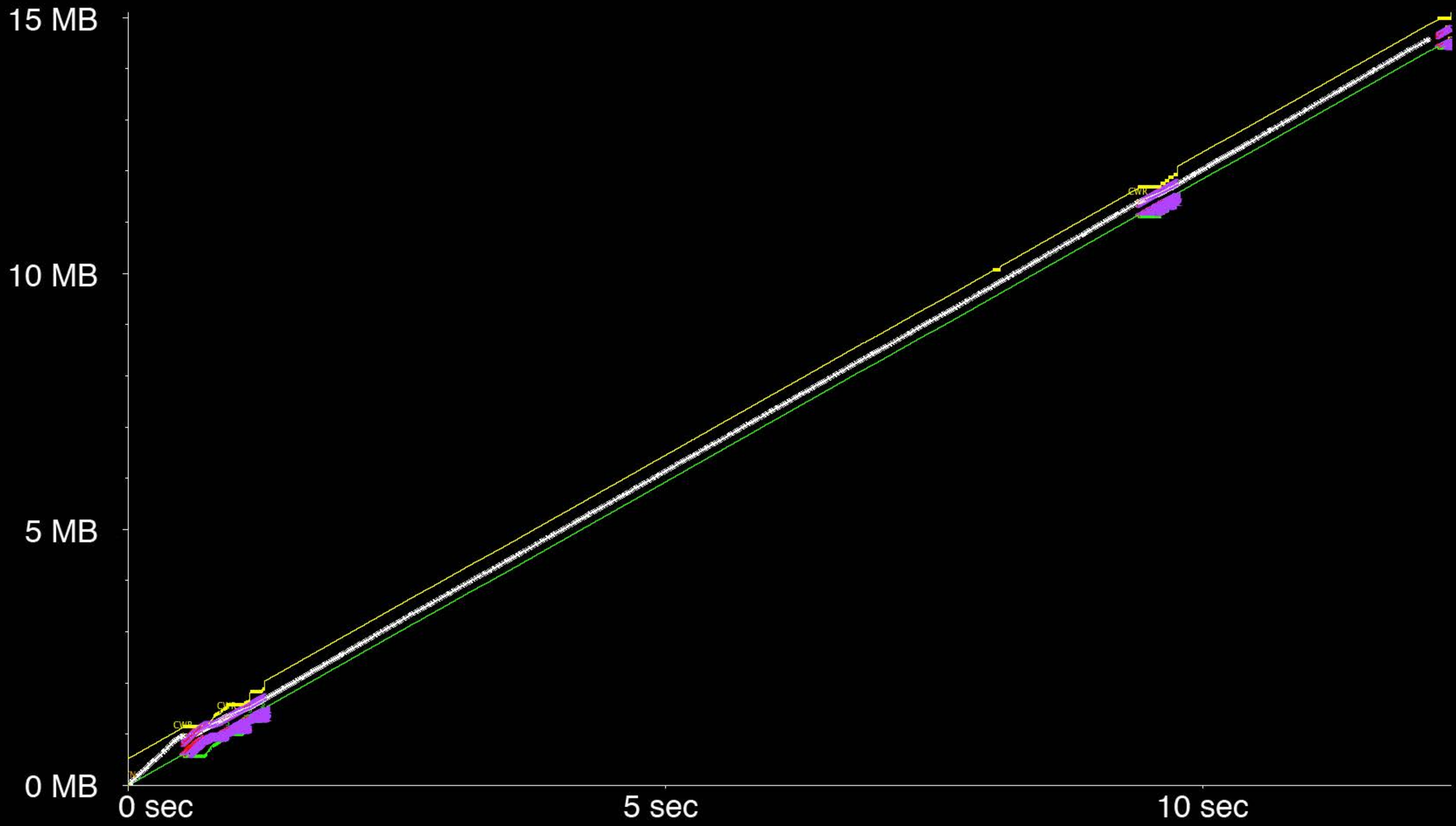
CoDel with ECN

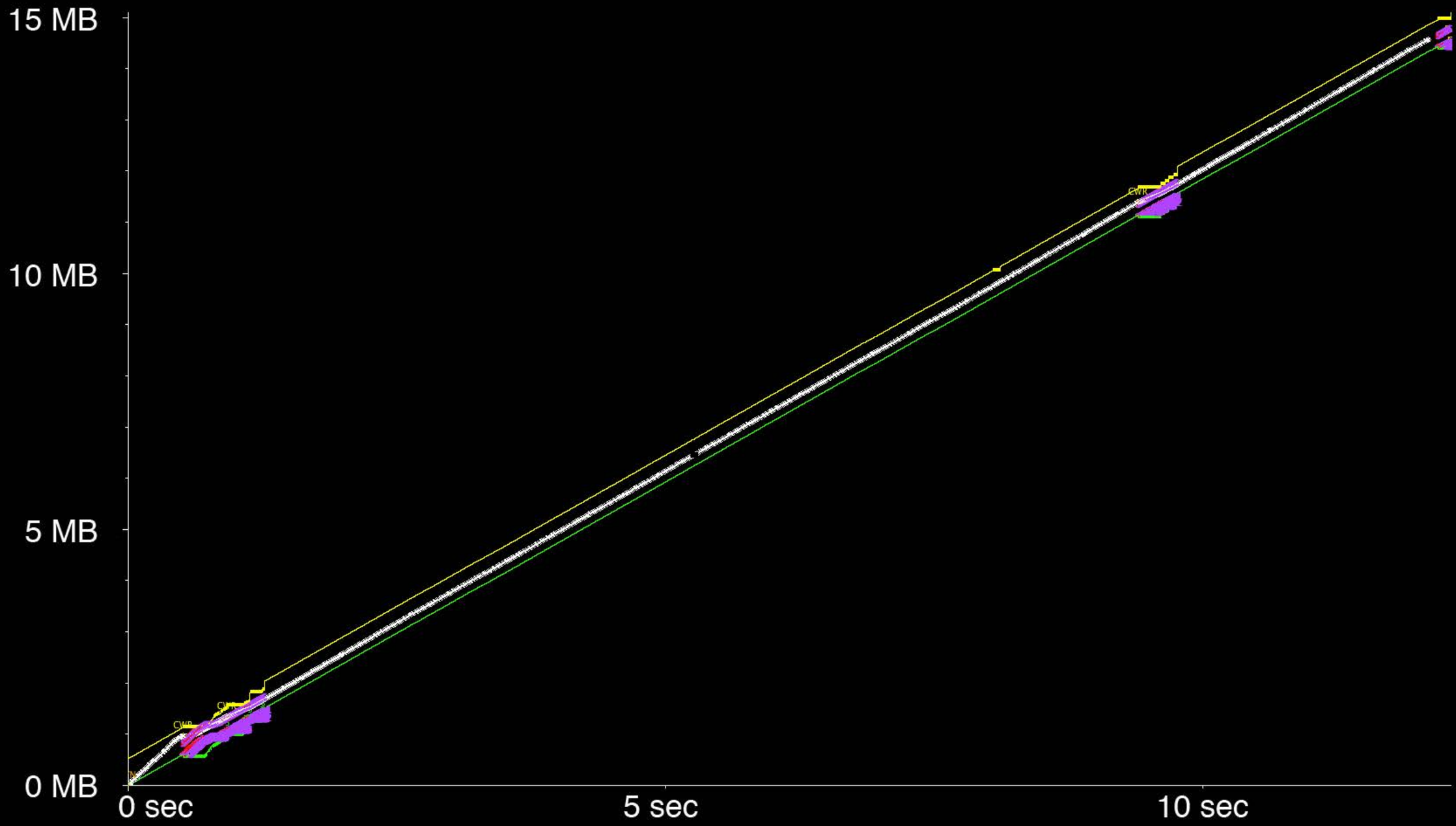
Gateway Device: CeroWRT 3.10.18-1

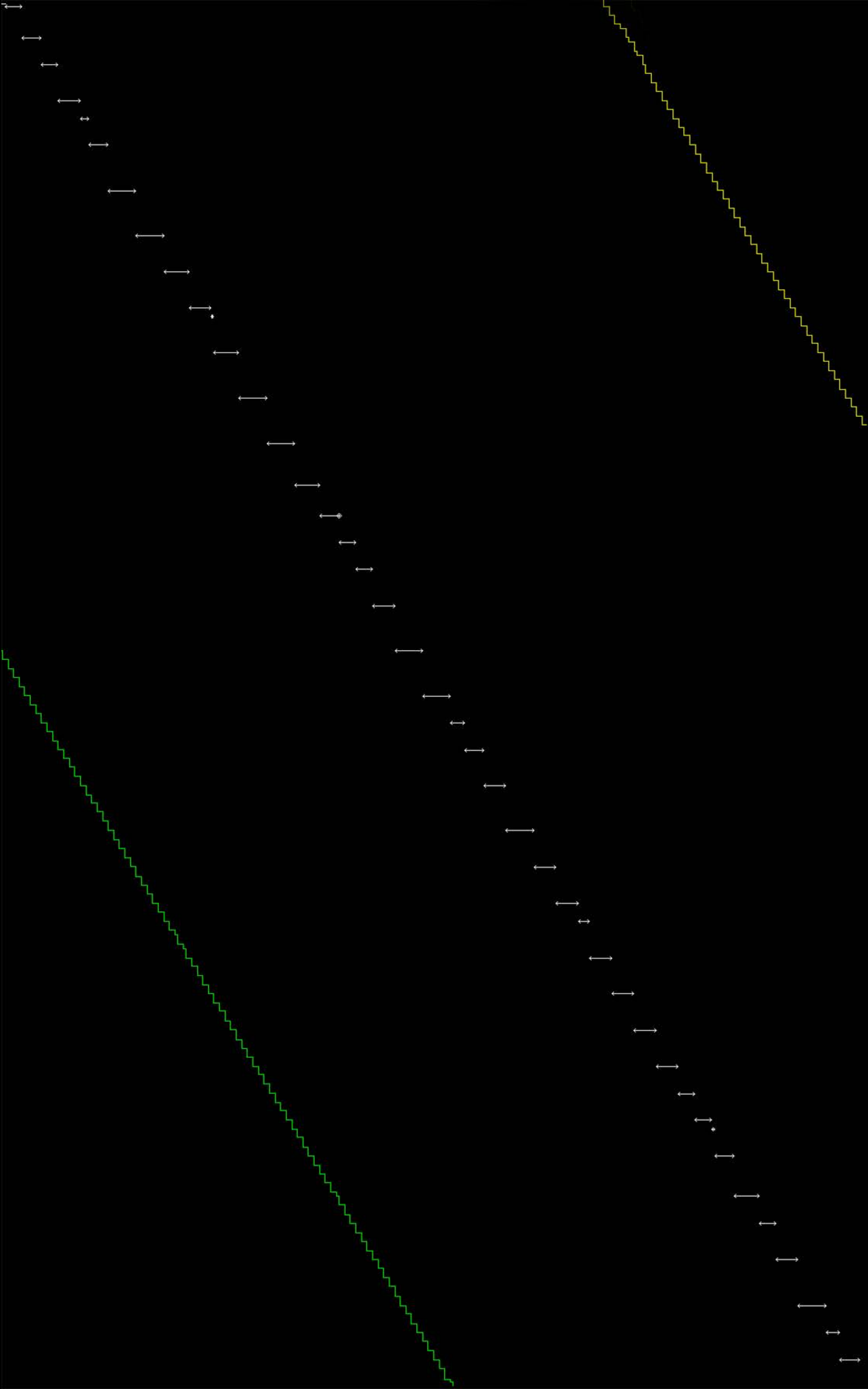
(< 1 ms intrinsic delay, so any delay is self-induced queueing delay)

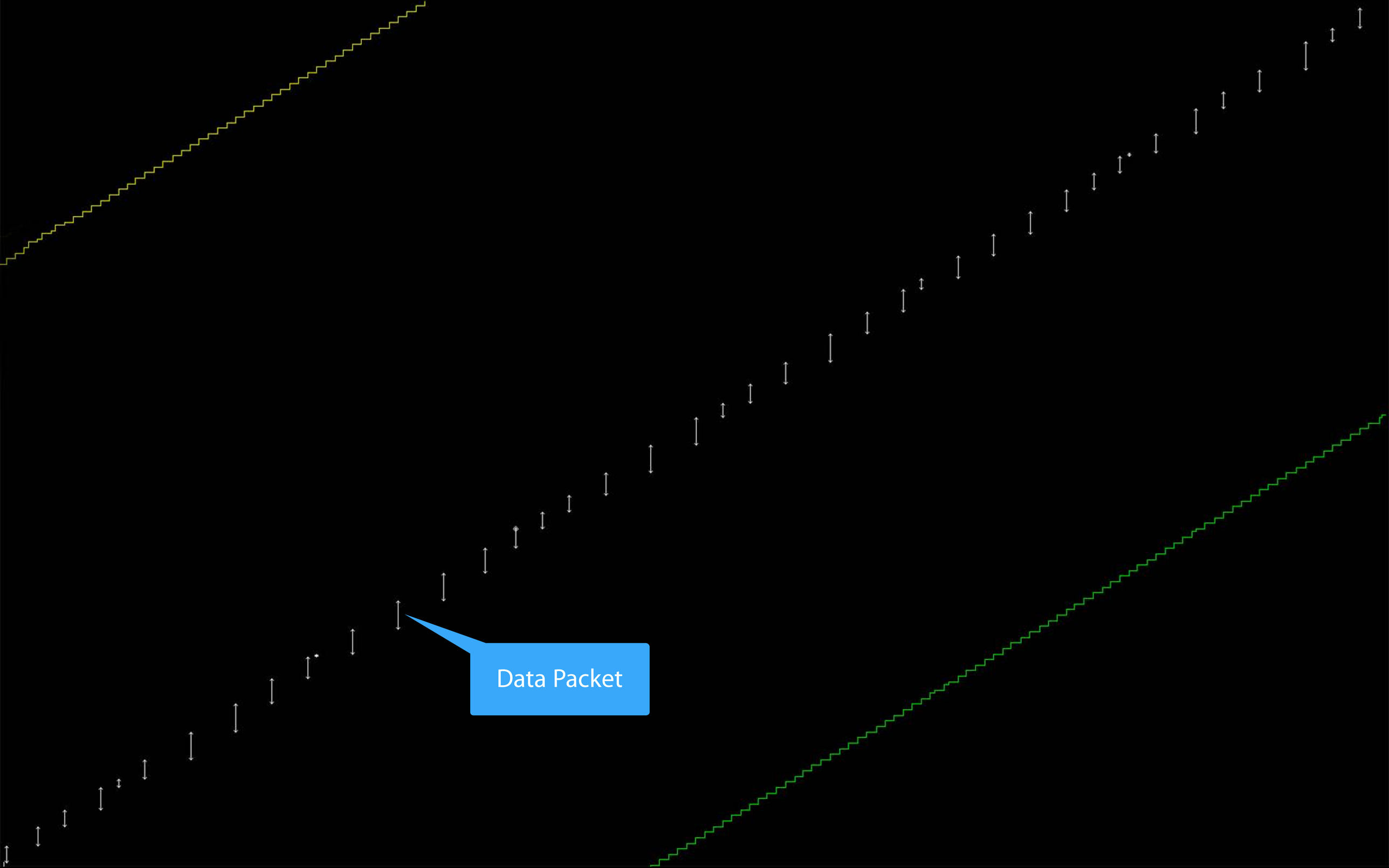
tcptrace

http://www.tcptrace.org/


15 MB
10 MB
5 MB
0 MB
0 sec
5 sec
10 sec

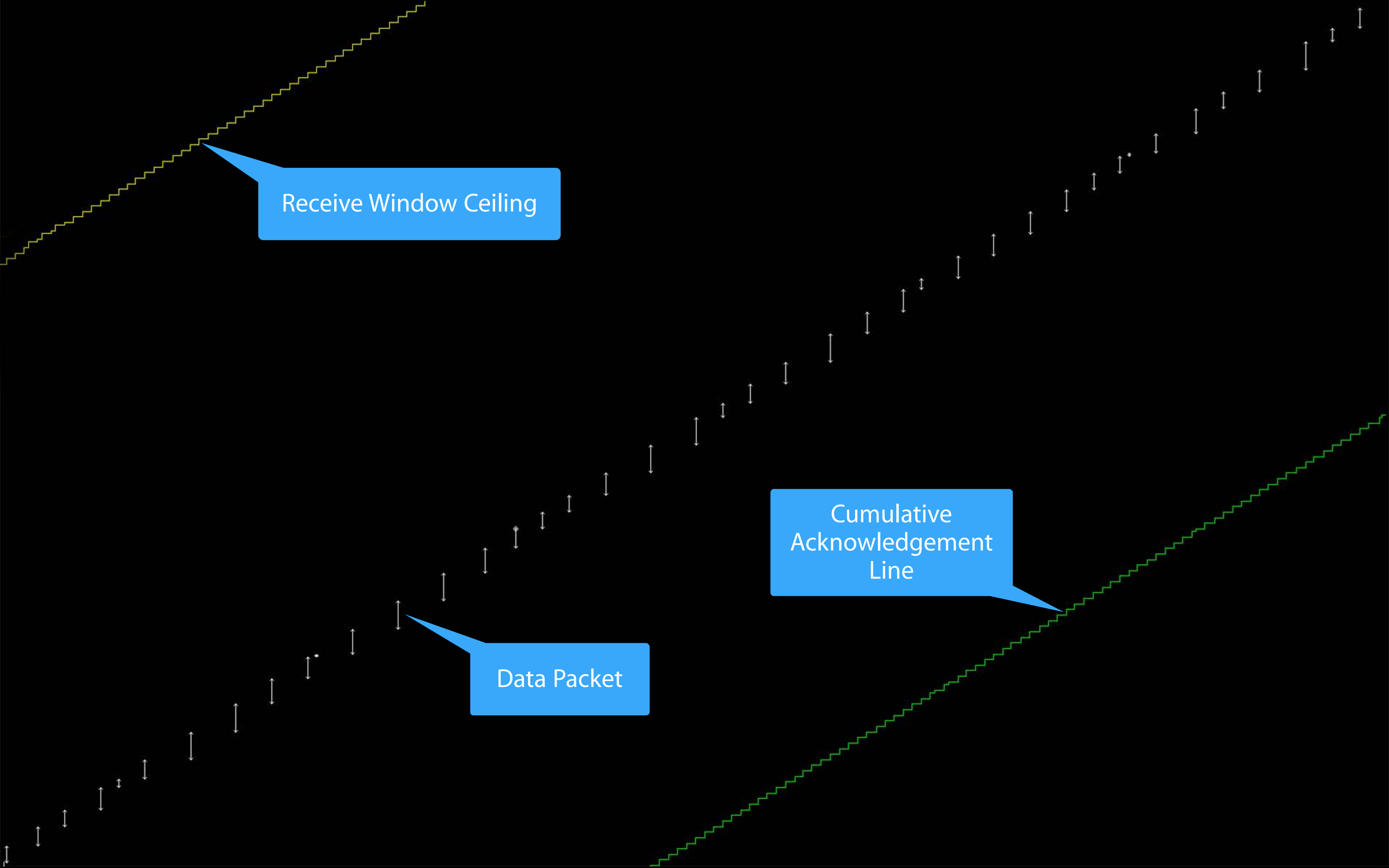

15 MB
10 MB
5 MB
0 MB
0 sec
5 sec
10 sec

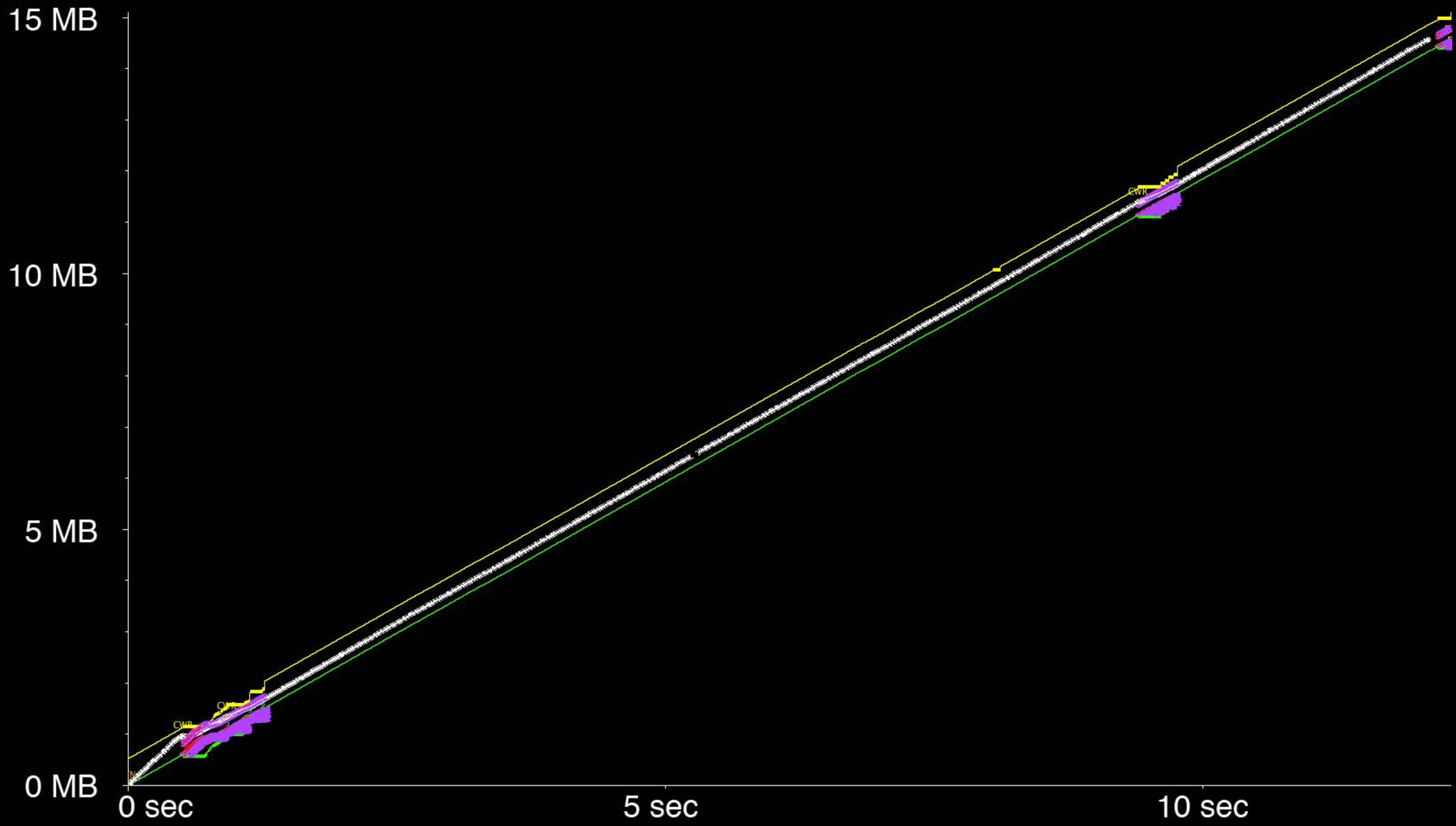


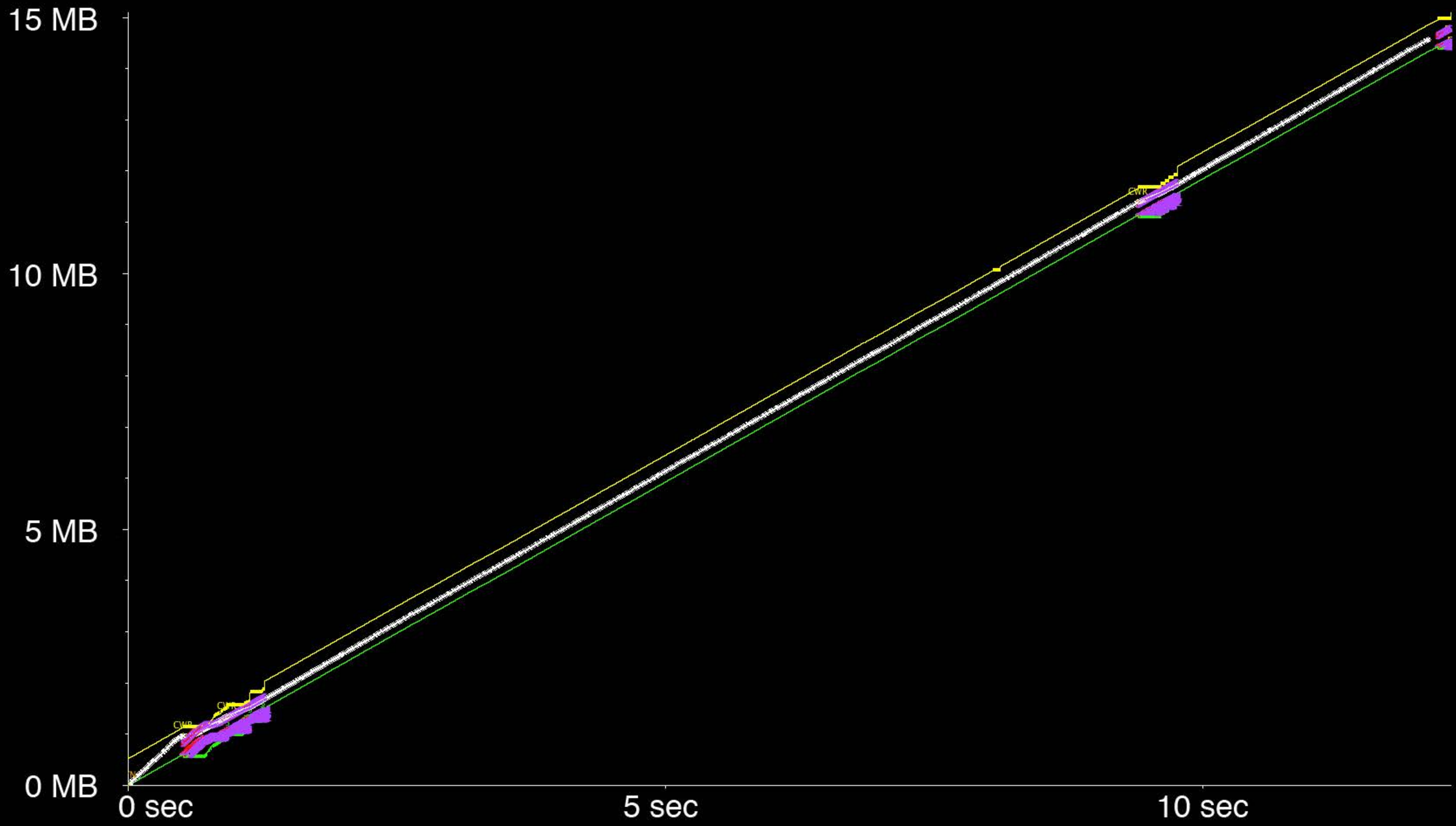

Data Packet

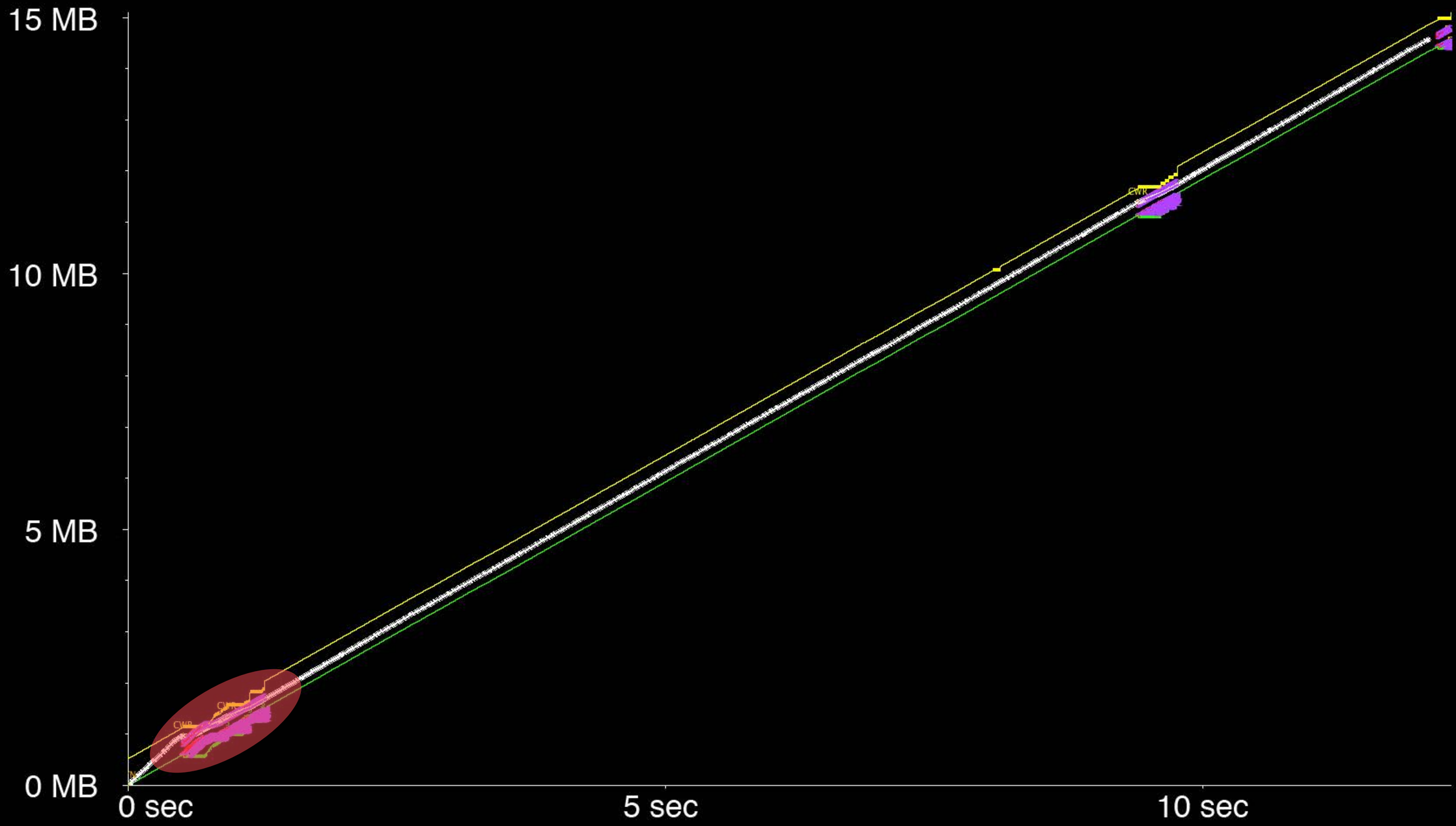
Cumulative Acknowledgement Line

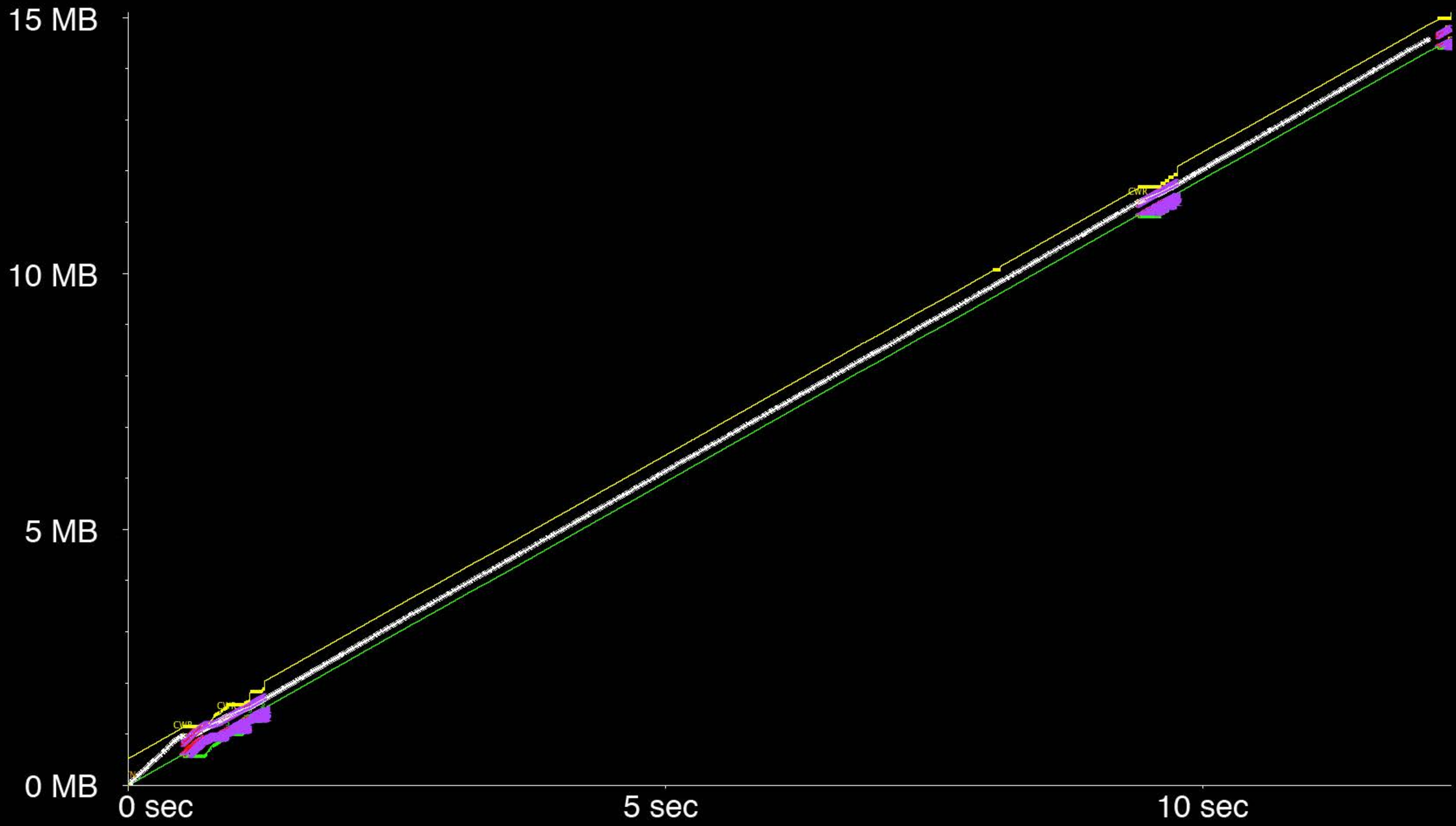
Data Packet


Receive Window Ceiling
Data Packet
Cumulative Acknowledgement Line


15 MB
10 MB
5 MB
0 MB
0 sec
5 sec
10 sec

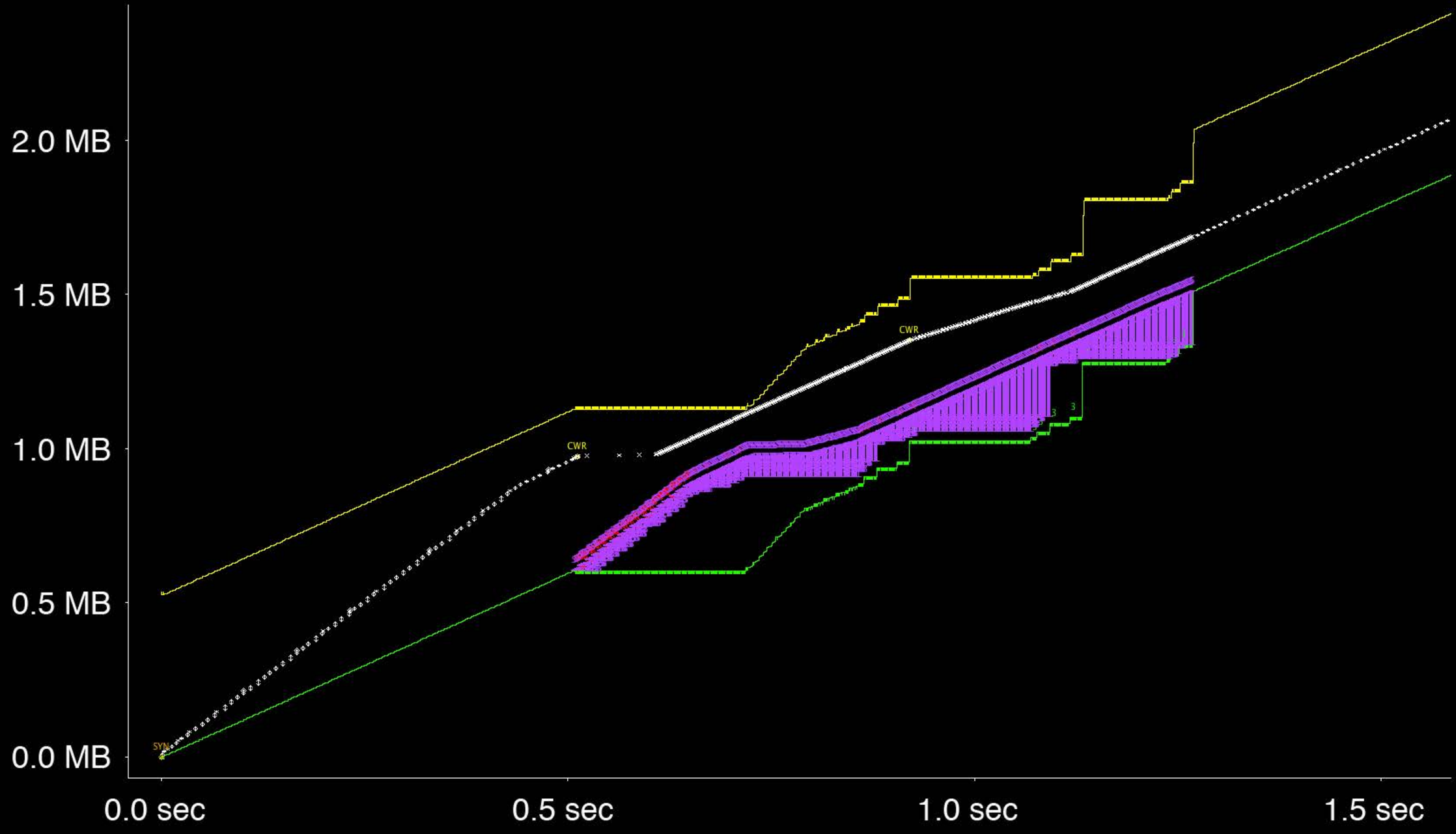

15 MB
10 MB
5 MB
0 MB
0 sec
5 sec
10 sec


15 MB
10 MB
5 MB
0 MB
0 sec
5 sec
10 sec

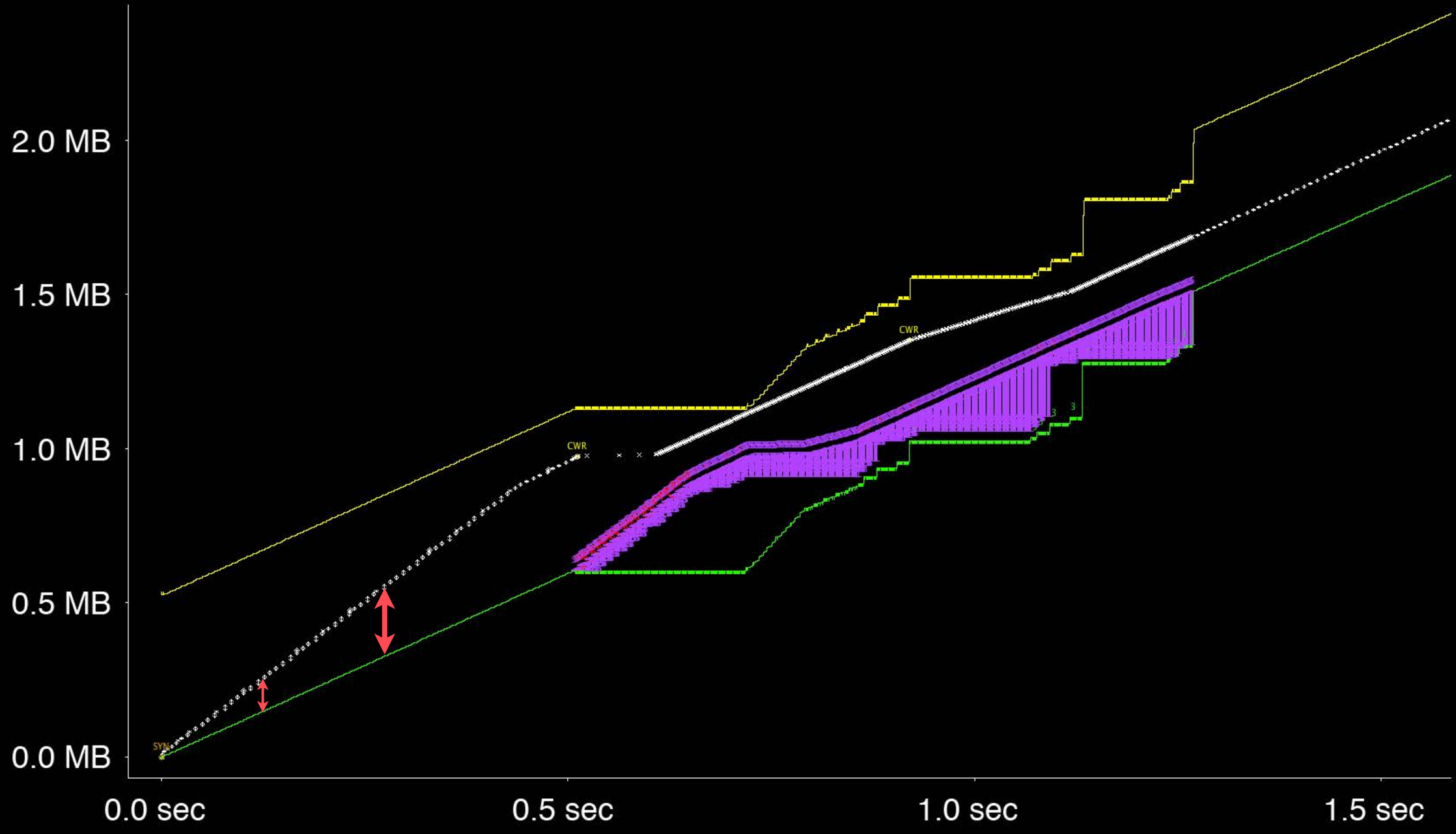

15 MB
10 MB
5 MB
0 MB
0 sec
5 sec
10 sec


CWP
TN
0 sec

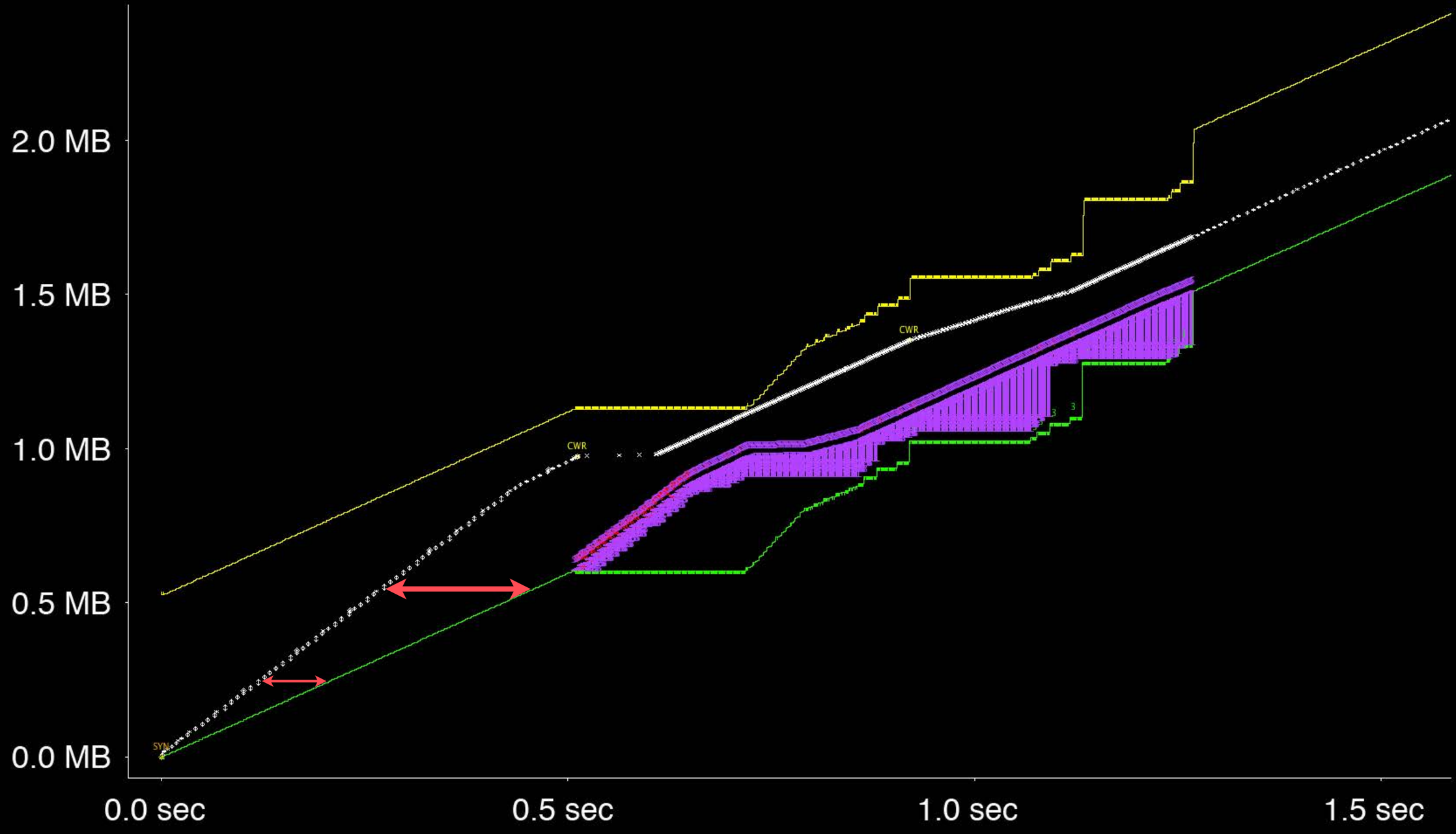
Standard FIFO Queue



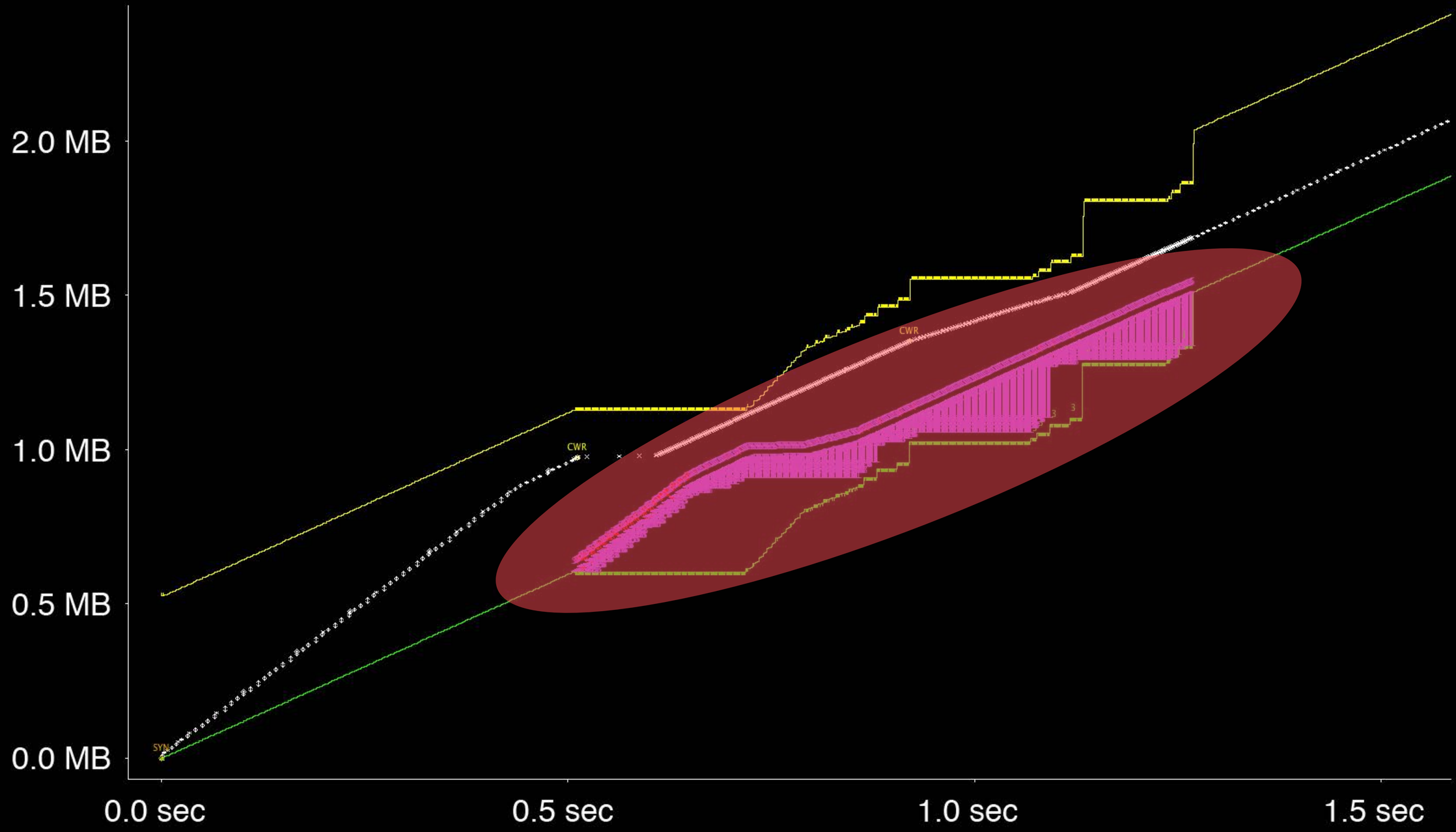
Standard FIFO Queue



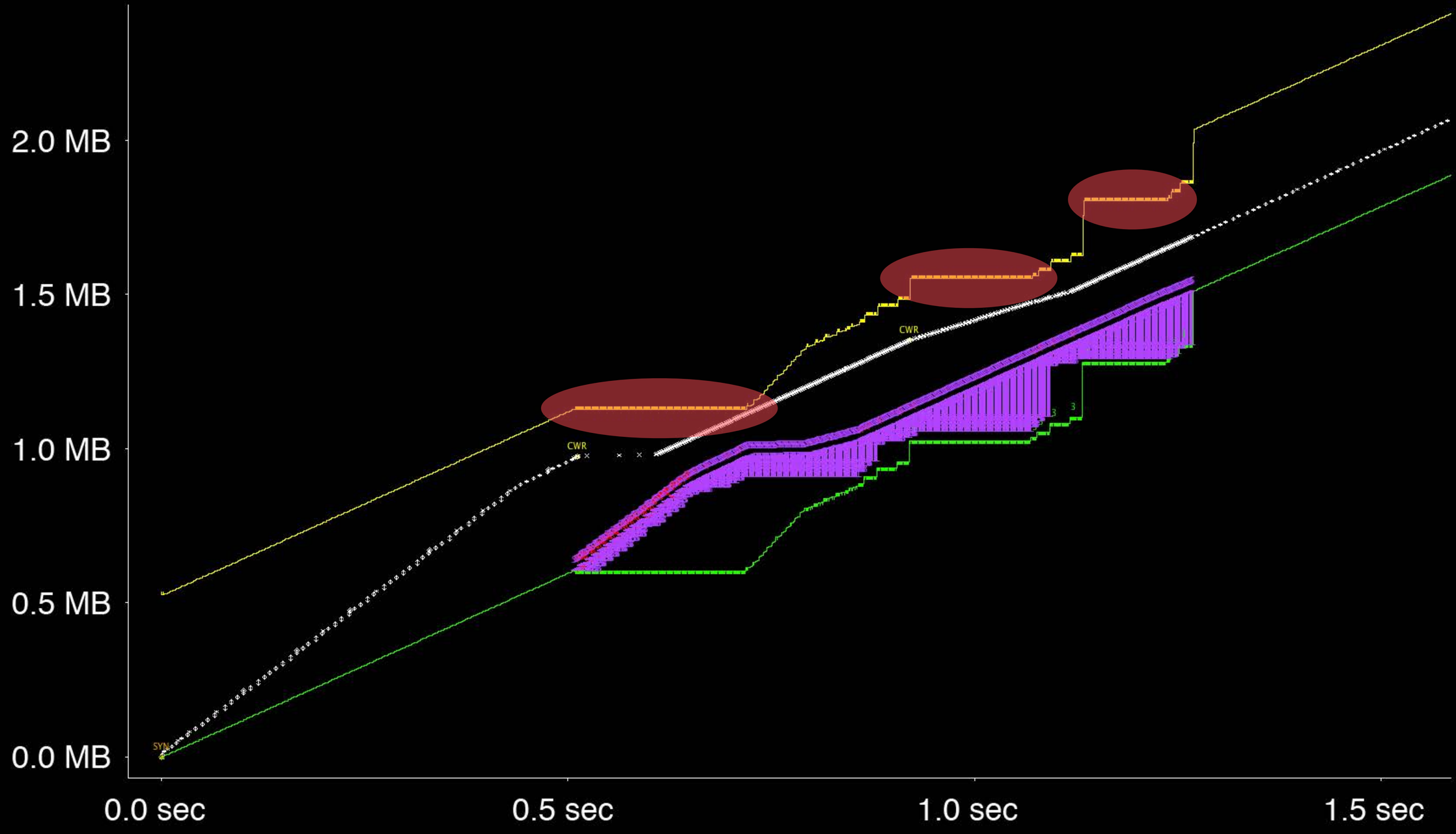
Standard FIFO Queue



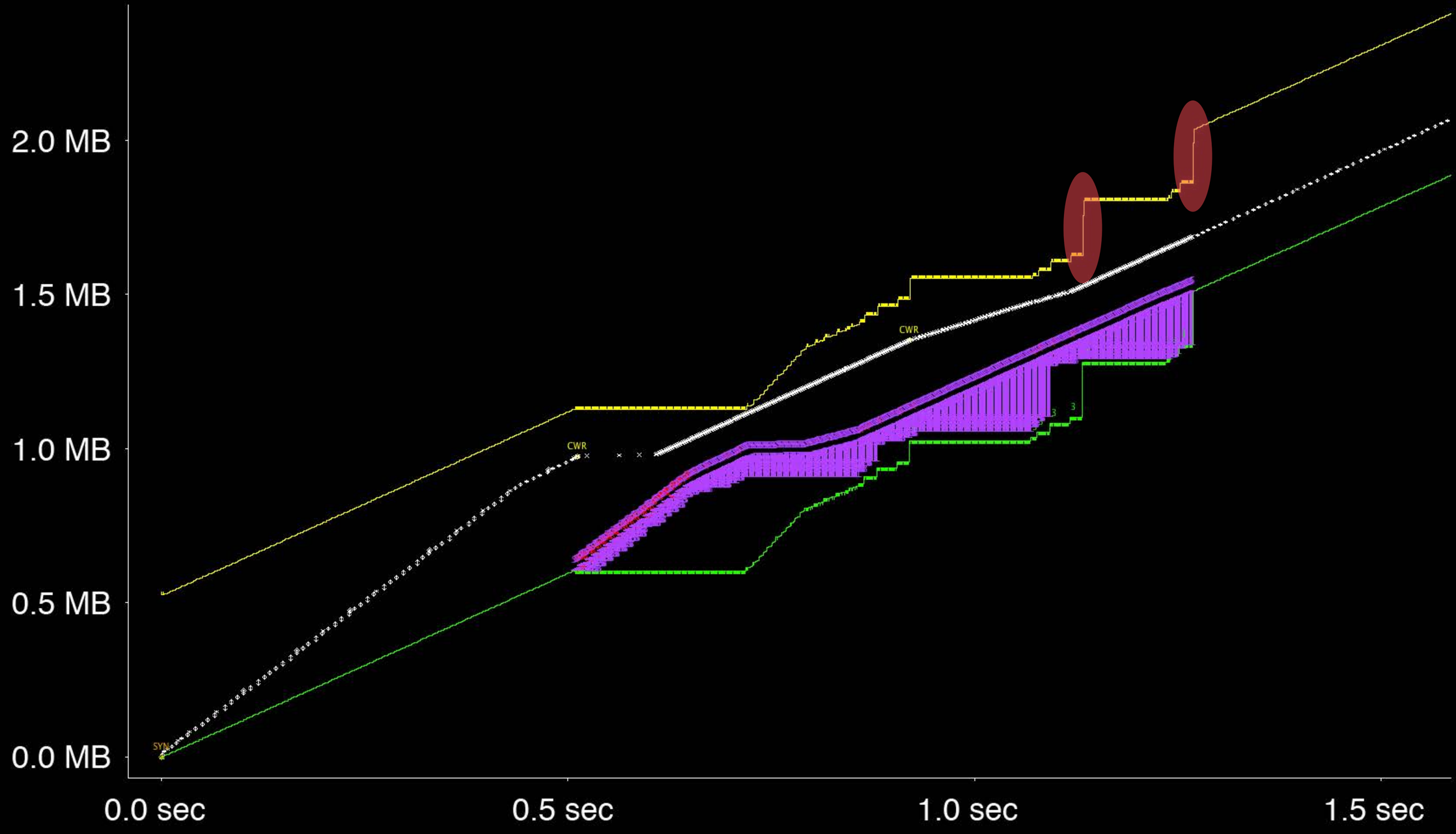
Standard FIFO Queue



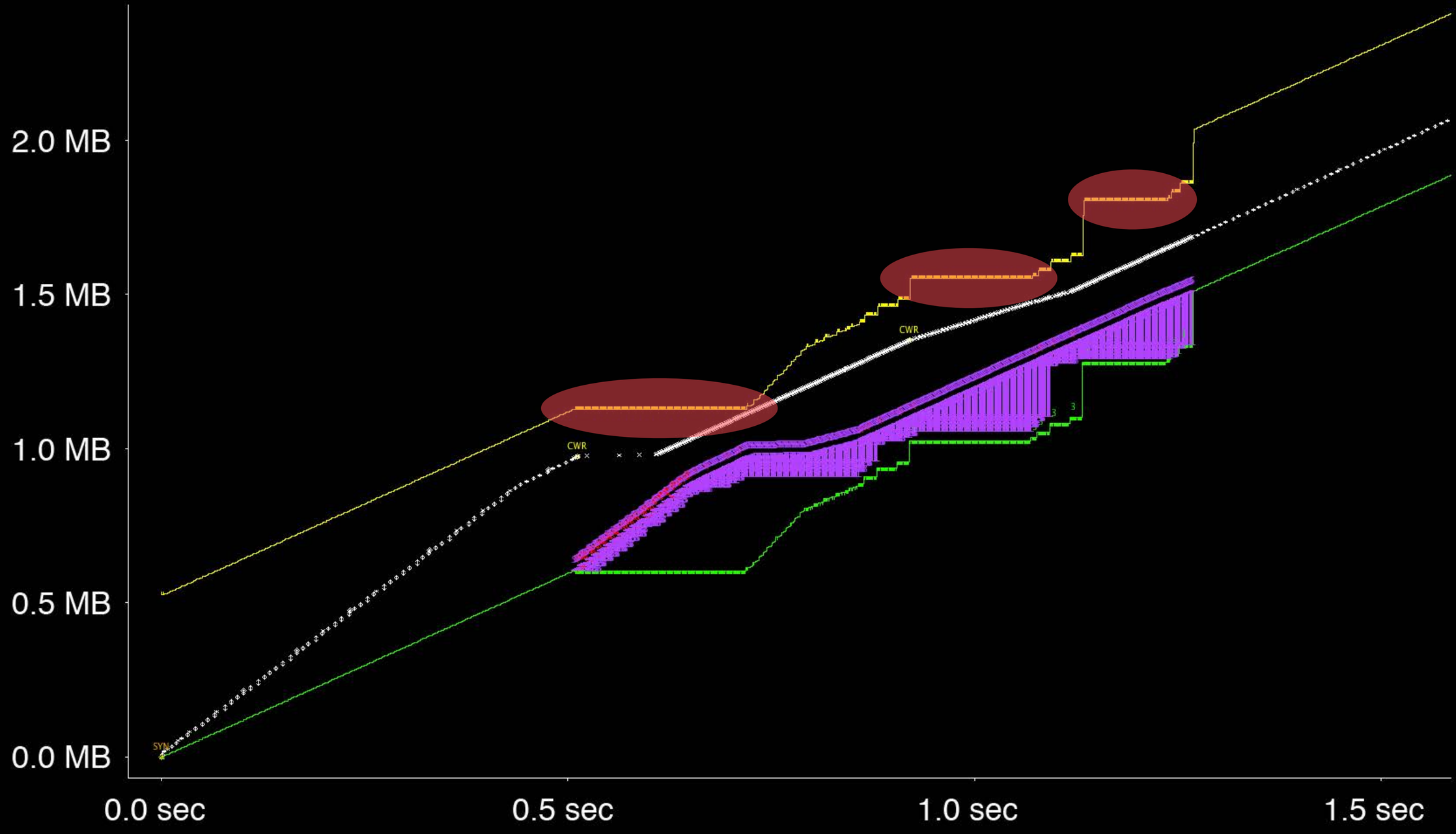
Standard FIFO Queue



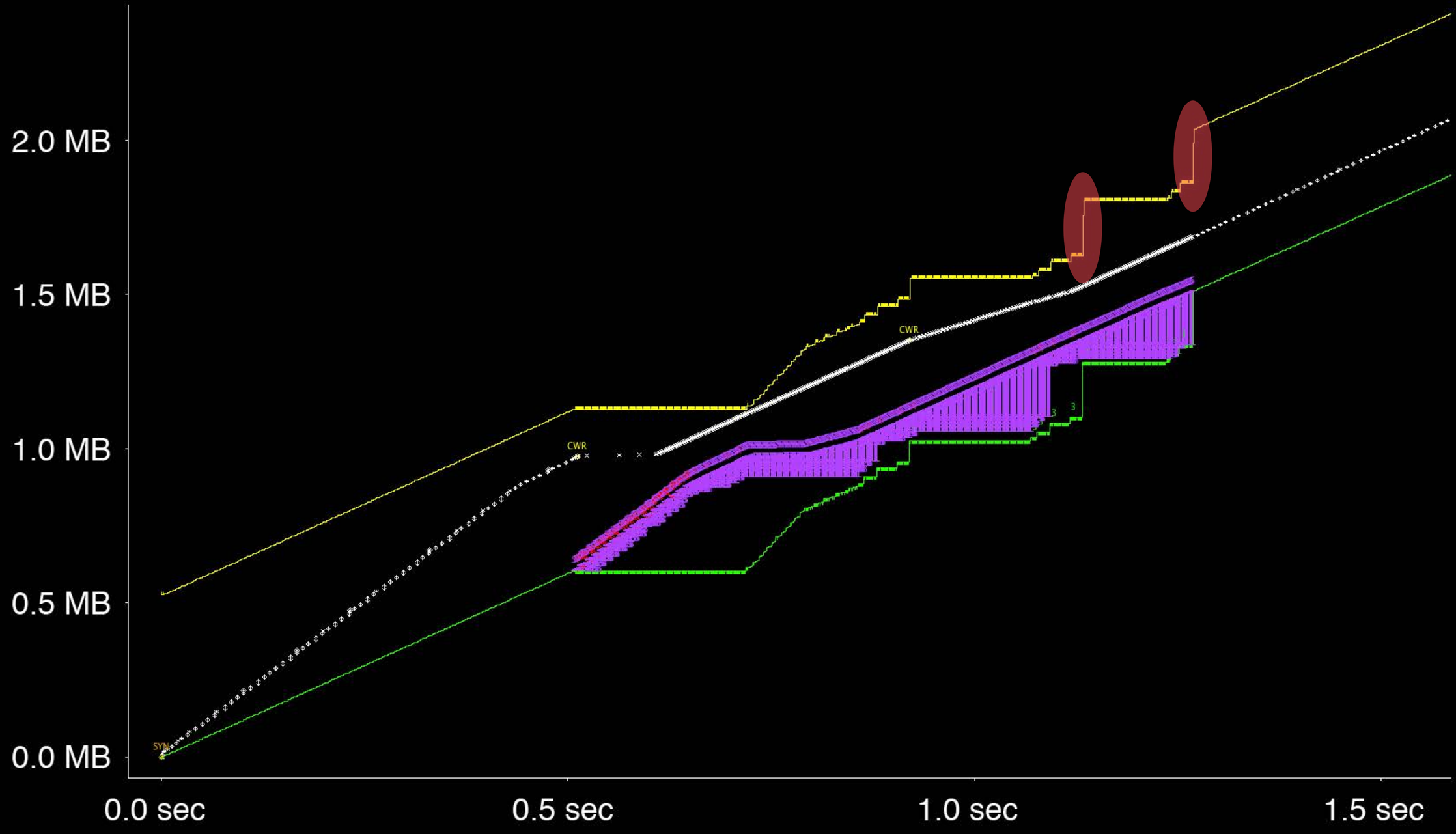
Standard FIFO Queue



Standard FIFO Queue



Standard FIFO Queue



Smart Queueing and ECN

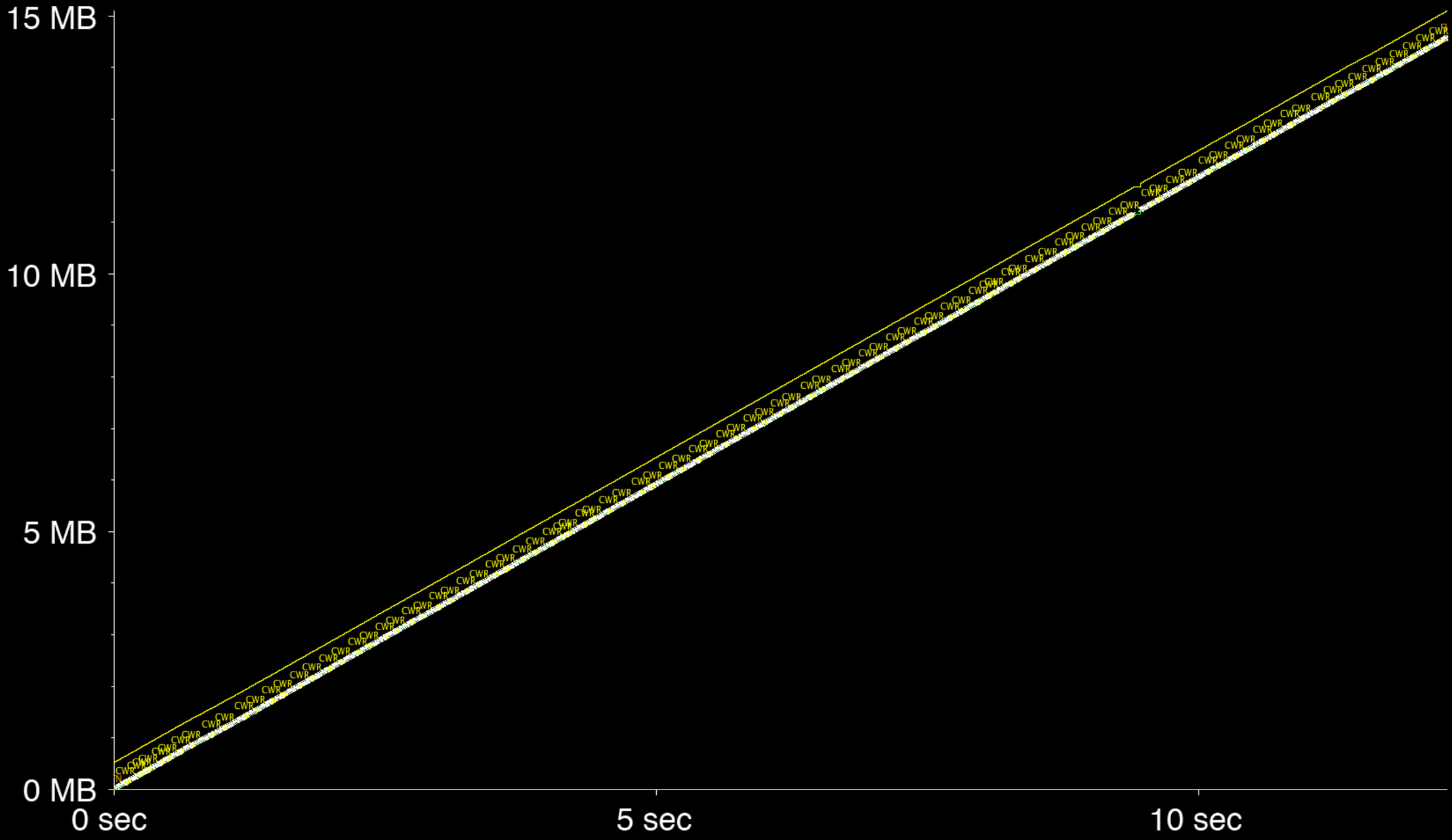
CoDel

- Controlled Delay queueing
- Limits Bufferbloat

Explicit Congestion Notification

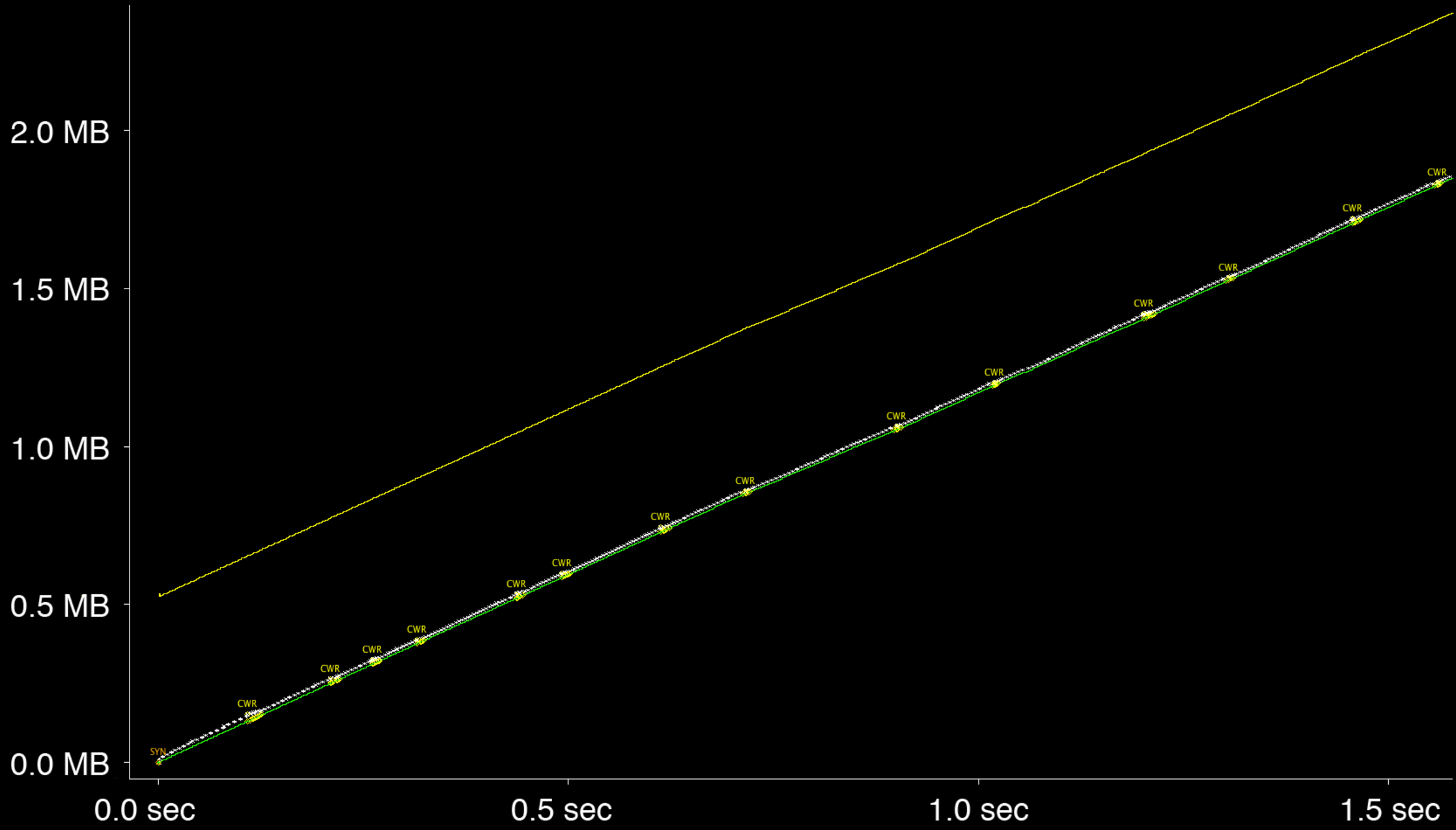
- Signals congestion by marking packets instead of discarding
- Available in OS X, iOS, Windows, Linux, etc.

CoDel with ECN

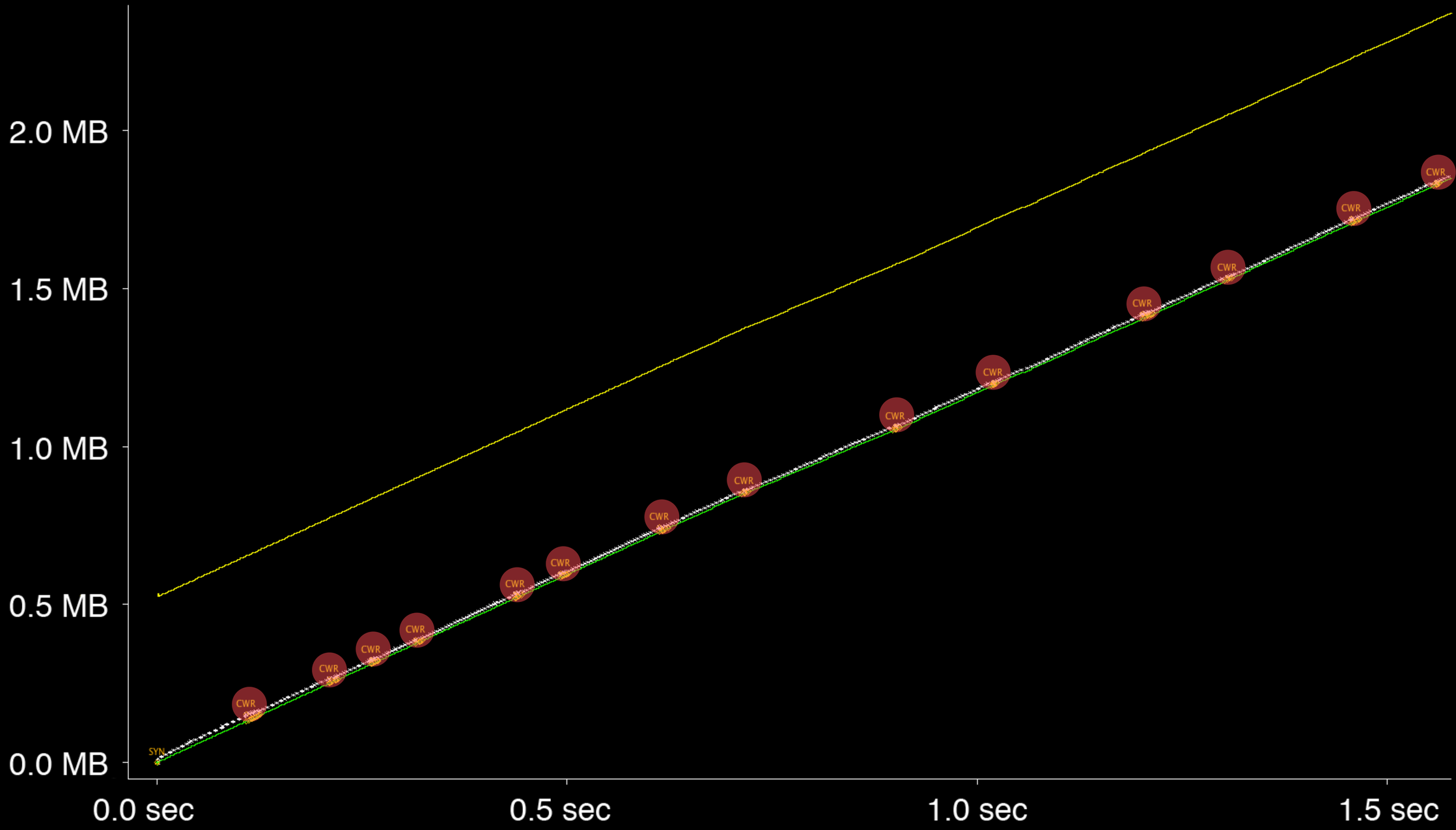



CWR
CWR
CWR
CWR
CWR
CWR
CWR
CWR
CWR
CWR
CWR
CWR
CWR
CWR
CWR
N

CoDel with ECN



CoDel with ECN



Conclusions

CoDel (or similar Smart Queue Management) helps

ECN helps

SQM+ECN really helps a lot

TCP for Streaming Video

Packet loss causes irregular data delivery to client

No problem for file transfer (e.g. sending an email)

Big problem for streaming video over TCP

- YouTube
- Netflix
- etc.

Changing Applications

Fixed data: Email, file transfer, etc.

- Fixed data
- Variable time (as fast as network can manage)

Adaptive data: Screen Sharing, Video Streaming, etc.

- Fixed time
- Variable data (as much as network can carry in allotted time)

Current State of ECN

Servers

- 56% of Alexa top million web sites already support ECN
- http://wan.poly.edu/pam2015/papers/4.pdf

Clients

- Routers aren't doing marking
- Some routers might drop the packets—small risk; no reward

Routers

- Clients aren't requesting ECN
- Enabling ECN might expose code bugs—small risk; no reward

Apple Is Taking the Initiative

ECN now enabled in OS X 10.11 and iOS 9

Test on your own home and work networks

Report bugs to Apple

We could have a billion iOS devices using ECN!

Finally, an incentive for ISPs to start offering ECN packet marking

All apps get this for free

Delay Reduction

Reliable Network Fallback

Reduce Connection Setup Stalls

Explicit Congestion Notification

Reduce Network Delays

TCP_NOTSENT_LOWAT

Reduce Sender-Side Delay

TCP Fast Open

Screen Sharing

Screen Sharing to home Mac over DSL

5 Mb/s downlink, 500 kb/s uplink

3-second delay on Screen Sharing

But ping time is 35 ms

Huh?

Socket Send Buffer

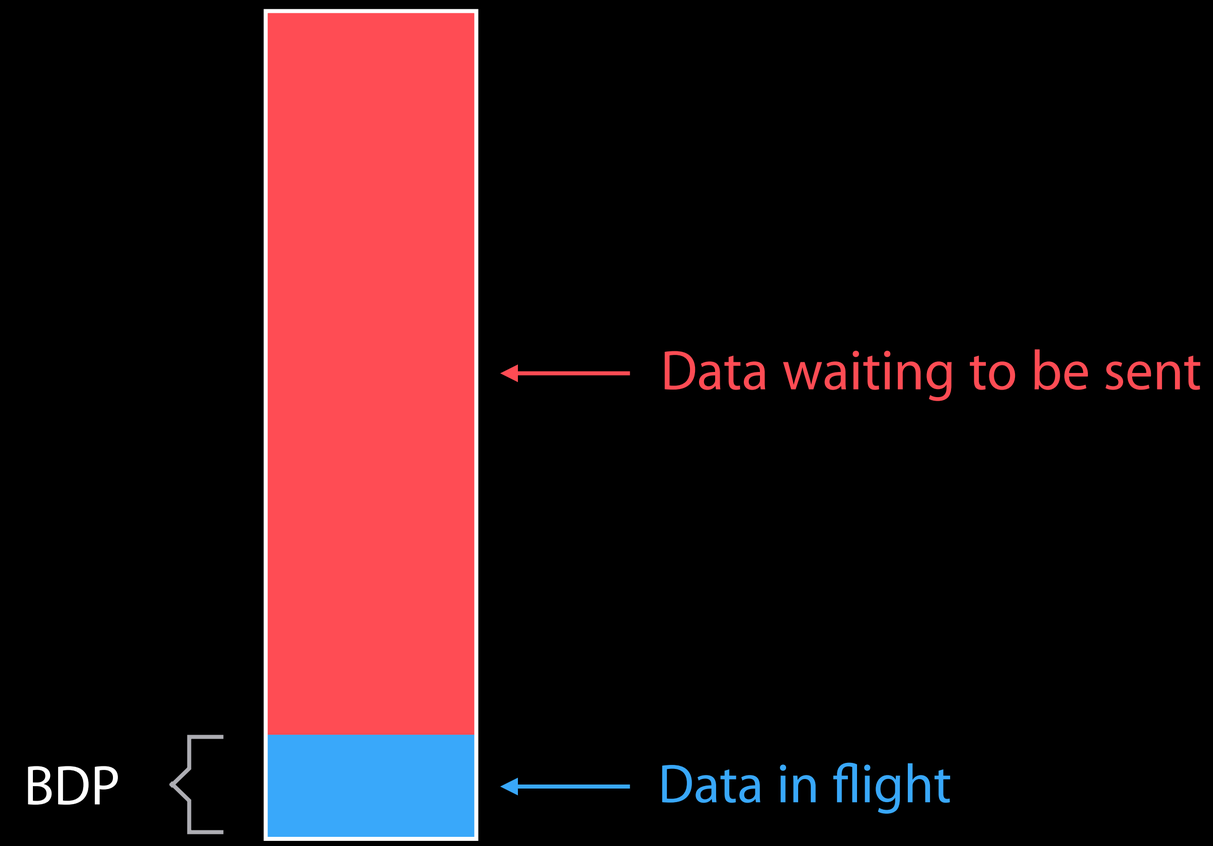
Socket Send Buffer is 128 kilobytes

Need send buffer large enough to hold
Bandwidth-Delay Product (BDP)

Any additional buffering just adds extra delay

At approximately 50 kB/sec transfer rate
128 kilobytes = 2.5 seconds of delay

Socket Send Buffer



Socket Send Buffer

At approximately 50 kB/sec transfer rate

128 kilobytes = 2.5 seconds

Delay is in host, not just the network

Do screen frames have to be aged in oak barrels before they're fit for consumption?

TCP_NOTSENT_LOWAT

setsockopt(skt, IPPROTO_TCP, TCP_NOTSENT_LOWAT, &threshold, sizeof(threshold));

Socket Send Buffer remains at 128 kilobytes

But kevent() doesn't report socket as writable until the unsent TCP data drops below specified threshold (typically 8 kilobytes)

Application then writes next single semantic unit of data

TCP_NOTSENT_LOWAT

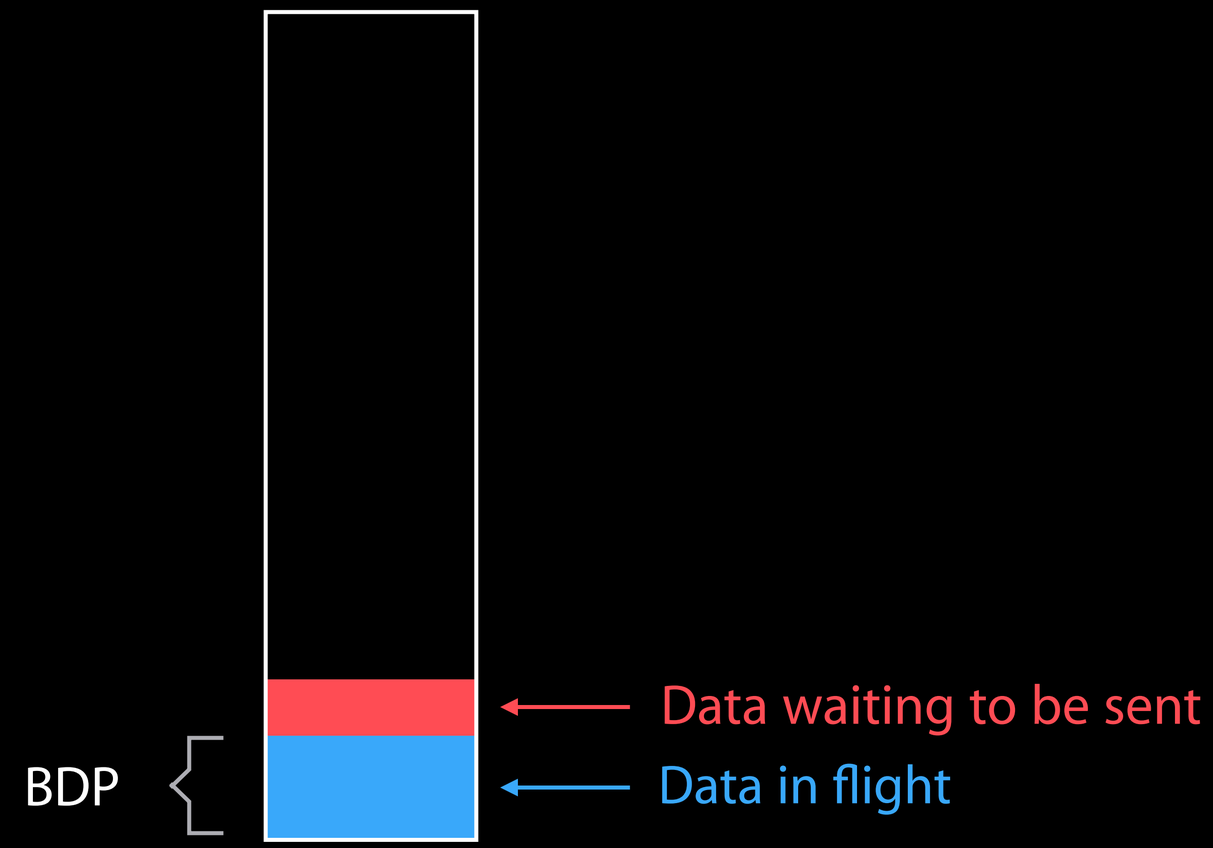
setsockopt(skt, IPPROTO_TCP, TCP_NOTSENT_LOWAT, &threshold, sizeof(threshold));

Socket Send Buffer remains at 128 kilobytes

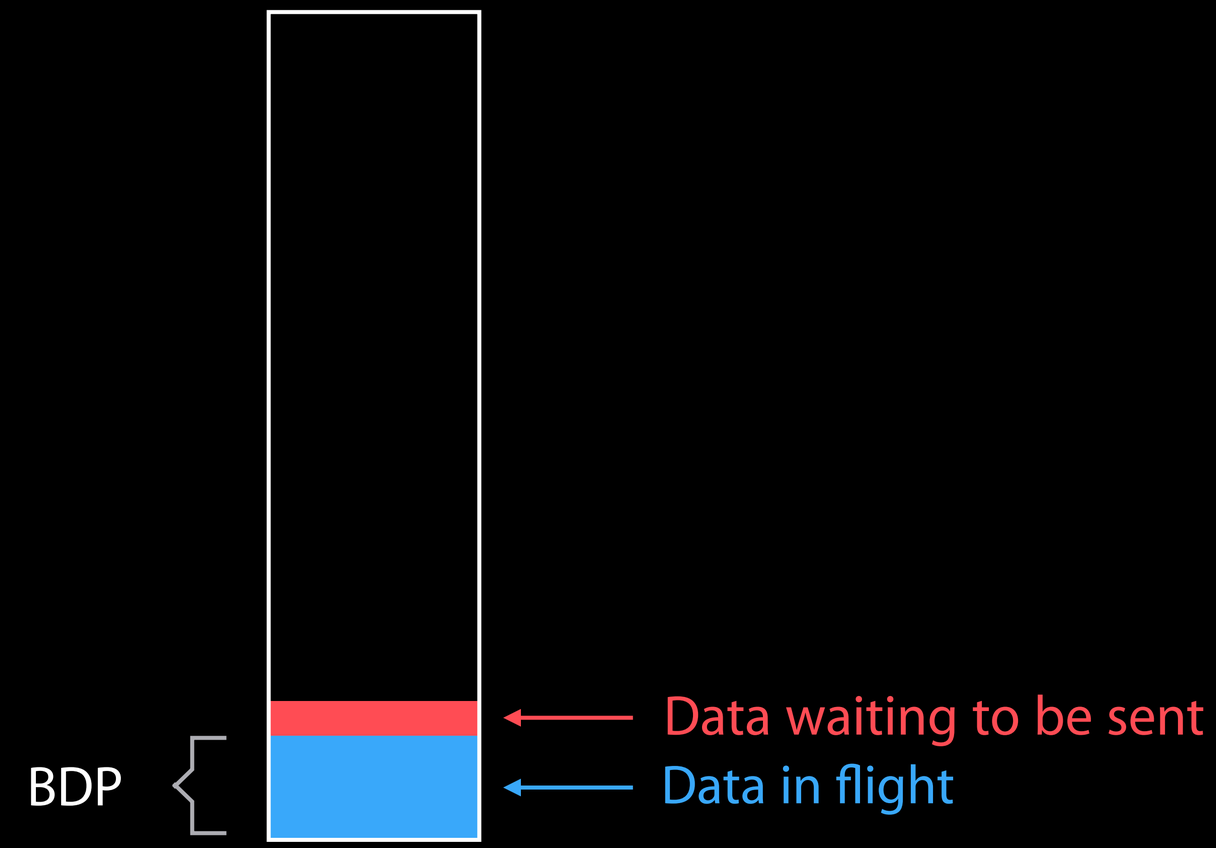
But kevent() doesn't report socket as writable until the unsent TCP data drops below specified threshold (typically 8 kilobytes)

Application then writes next single semantic unit of data

Socket Send Buffer

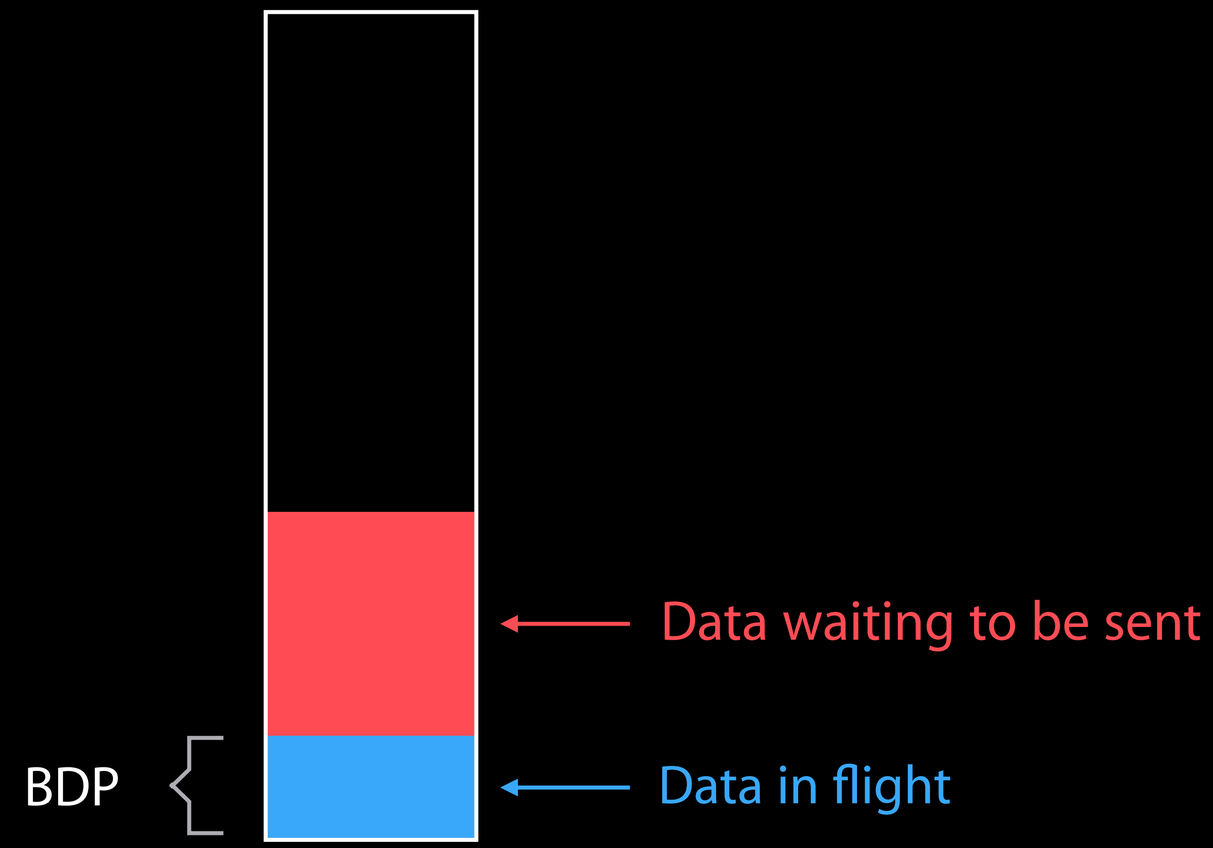


Buffer Reaches Threshold



Application Sends Next Chunk

Write One Atomic Semantic Unit



Demo

TCP_NOTSENT_LOWAT

Screen Sharing now using this in 10.10.3 and later

Used by AirPlay

Available in Linux too, for your server software

Good for All Applications

Obvious benefit for "real time" applications

- But *all* applications benefit

Use the NSURLSession and CFNetwork-layer APIs

When runloop reports socket is writable:

- Write a *single* semantic atomic chunk
- Don't loop until EWOULDBLOCK

Delay Reduction

Reliable Network Fallback

Reduce Connection Setup Stalls

Explicit Congestion Notification

Reduce Network Delays

TCP_NOTSENT_LOWAT

Reduce Sender-Side Delay

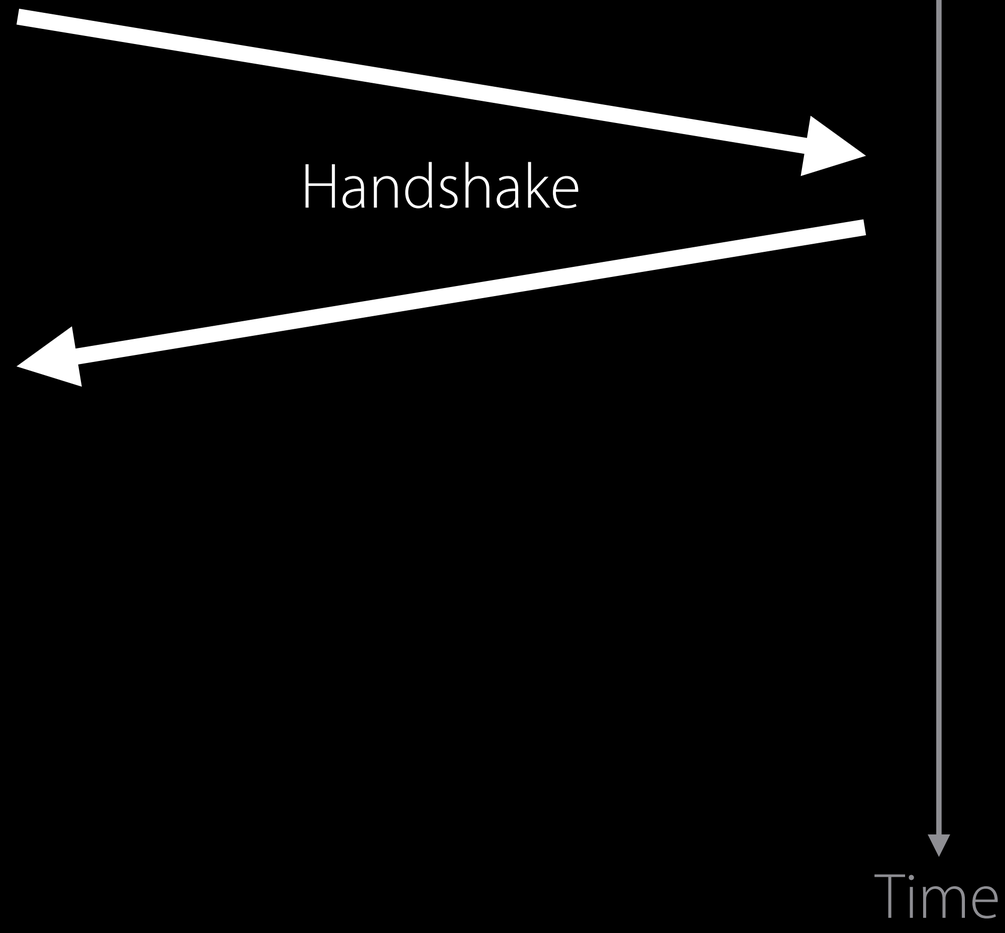
TCP Fast Open

Accelerating the TCP handshake

TCP Fast Open

Accelerating the TCP handshake

TCP handshake takes one round-trip-time

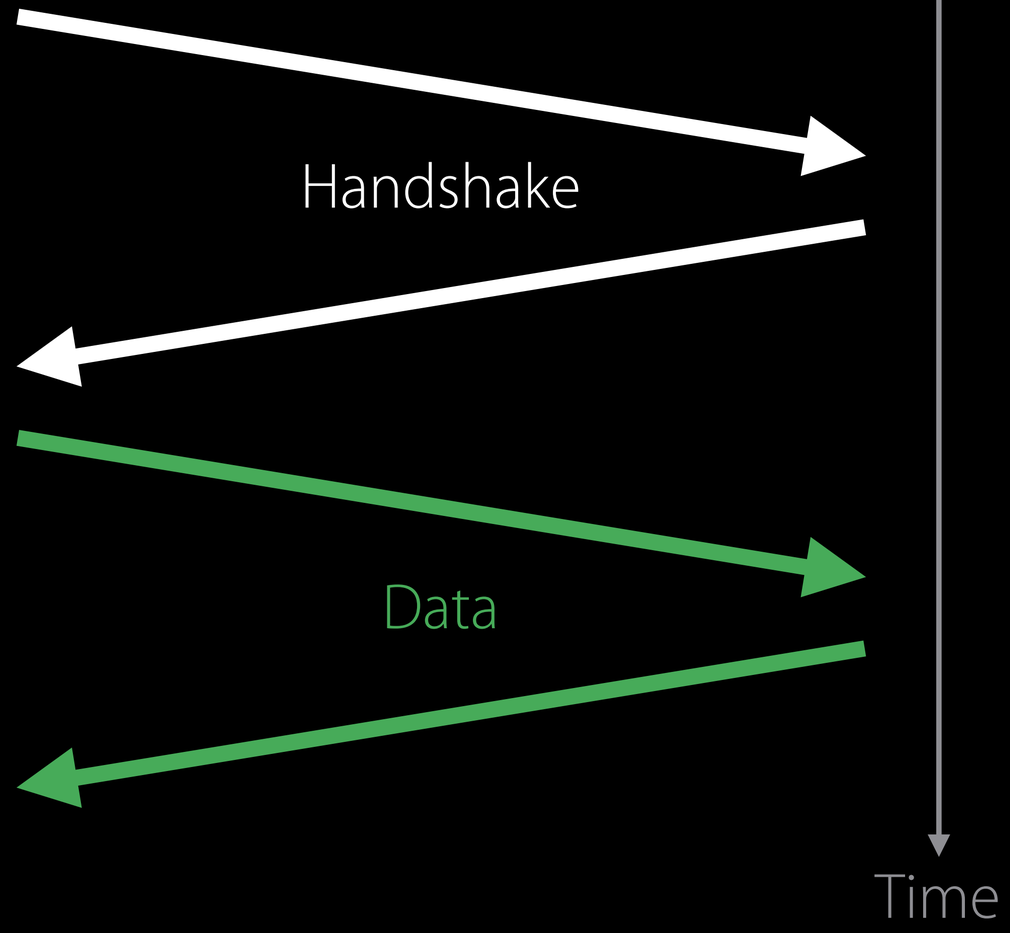


TCP Fast Open

Accelerating the TCP handshake

TCP handshake takes one round-trip-time

Data can only be sent afterwards

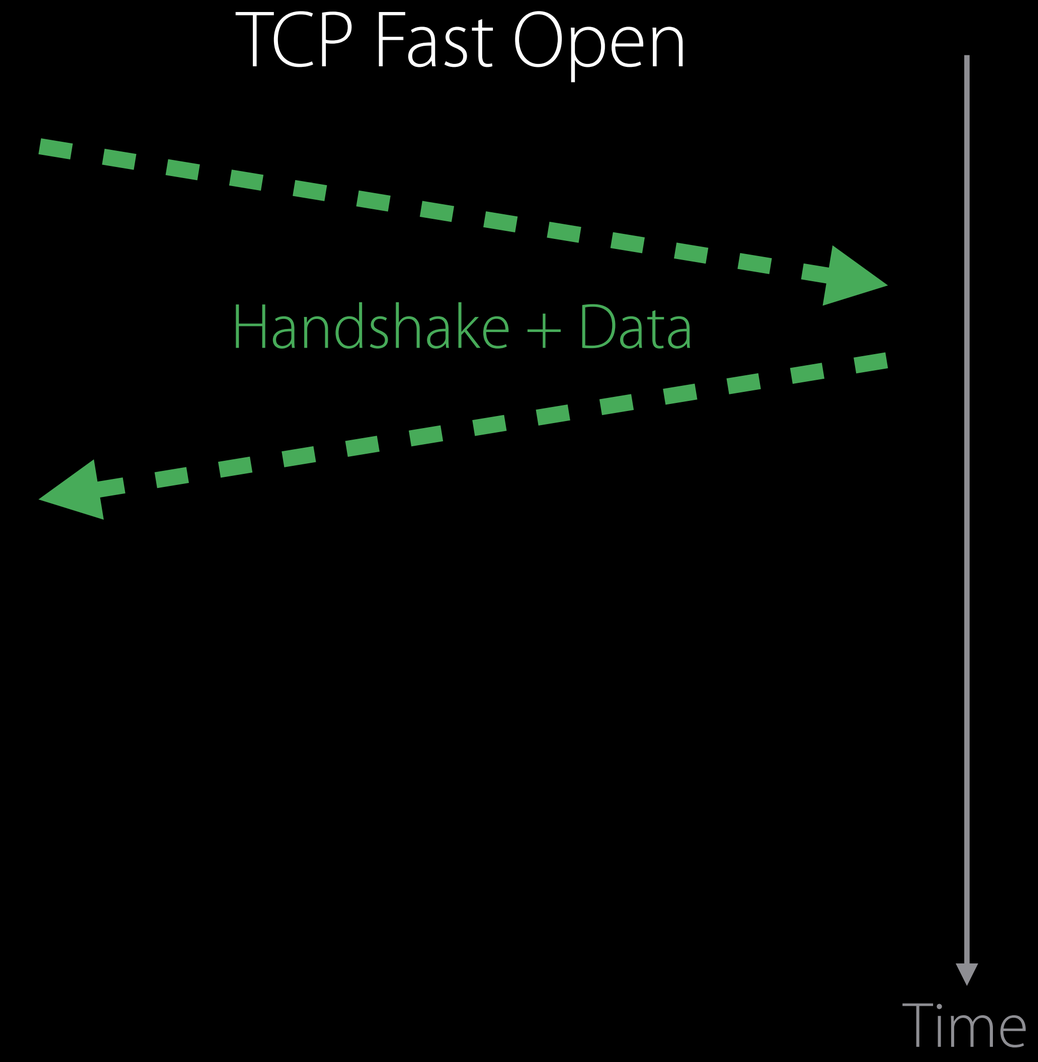


TCP Fast Open

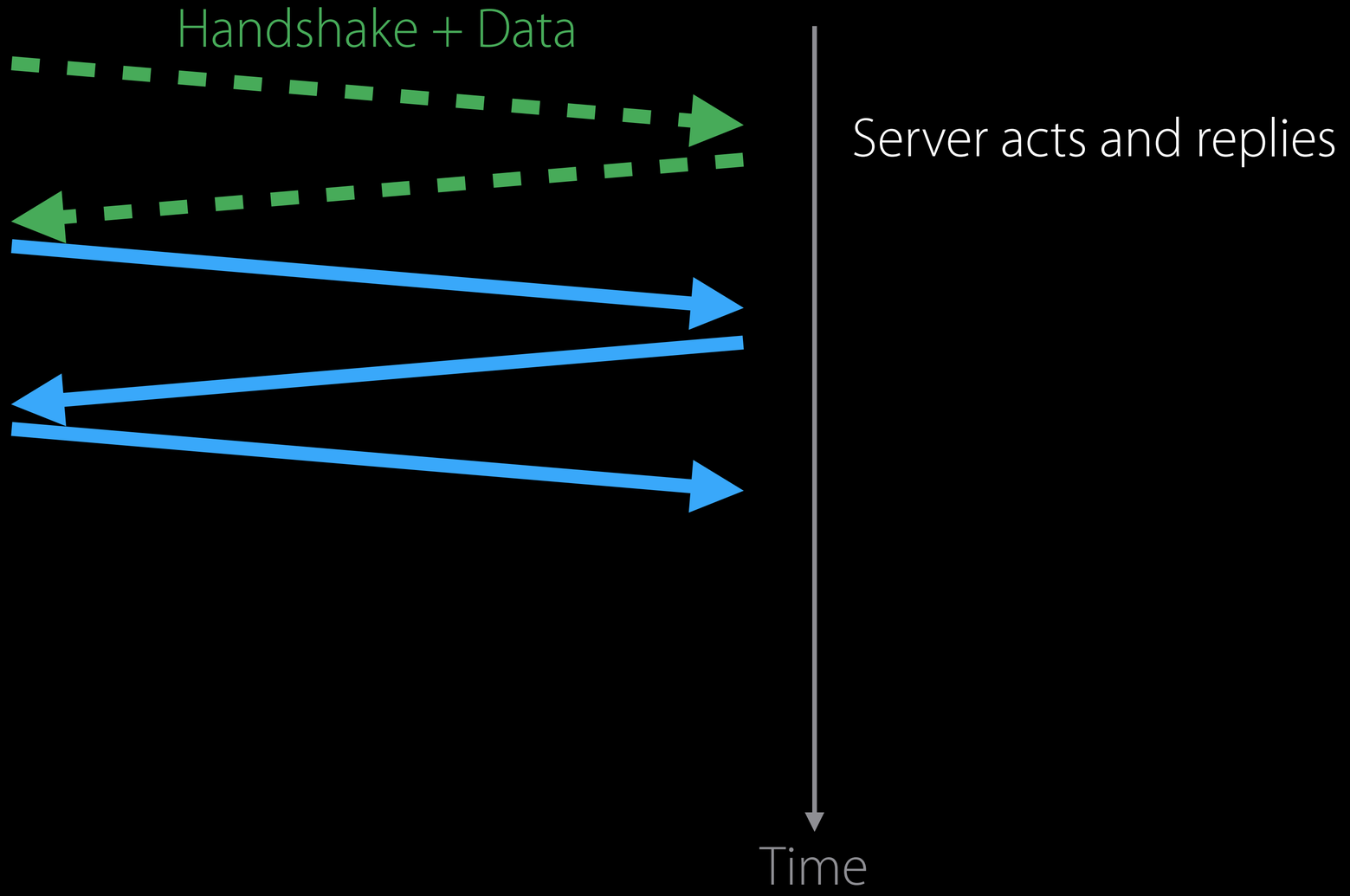
Accelerating the TCP handshake

TCP Fast Open

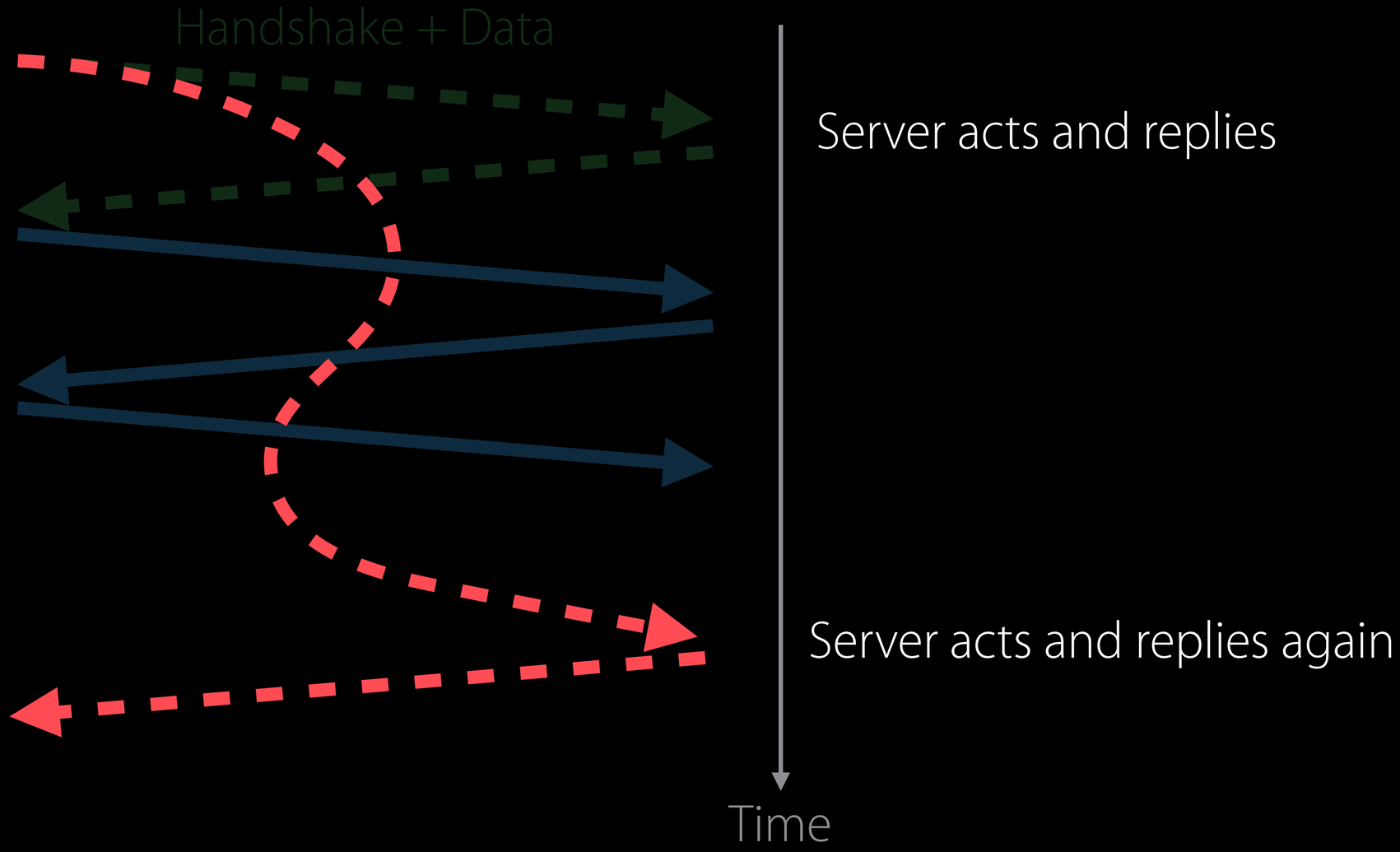
- Combines the handshake with data
- 50% latency reduction for short flows
- Secured through Cookie-exchange
- Only for "idempotent" data



TCP Fast Open Only for Idempotent Data



TCP Fast Open Only for Idempotent Data



TCP Fast Open

How to use it?

- Socket API
  - Using connectx() system call to combine handshake with data:

```
connectx(fd, ..., DATA_IDEMPOTENT | CONNECT_RESUME_ON_READ_WRITE, ...); // SYN delayed
write(fd, ...);                                                          // SYN goes out with first data segment
```

- Server-side
  - Must support TFO and application has to opt-in
  - iOS/OS X: Socket-option TCP_FASTOPEN
  - Linux (requires v4.1+)

Summary

Use NSURLSession and CFNetwork-layer APIs

Test on NAT64 + DNS64 network

Reliable Network Fallback

- Better Route notifications

Explicit Congestion Notification

TCP_NOTSENT_LOWAT

- Don’t over-stuff

TCP Fast Open technology preview

More Information

Documentation and Videos

Networking Programming Topics
https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/NetworkingTopics/Introduction/Introduction.html

CFNetwork
https://developer.apple.com/library/mac/documentation/Networking/Conceptual/CFNetwork/Introduction/Introduction.html

NSURLSession
https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/URLLoadingSystem/URLLoadingSystem.html#//apple_ref/doc/uid/10000165-BCICJDHA

More Information

Technical Support

Apple Developer Forums

http://developer.apple.com/forums

Developer Technical Support

http://developer.apple.com/support/technical

General Inquiries

Paul Danbold, Core OS Evangelist

danbold@apple.com

Related Sessions

Networking with NSURLSession	Pacific Heights	Thursday 9:00AM
What's New in Network Extension and VPN	Nob Hill	Friday 9:00AM

Related Sessions

Networking Lab	Frameworks Lab E	Friday 1:30PM

WWDC15