Square Pegs in a Round Pipe: Wire-Compatible Unordered Delivery In TCP and TLS

Michael F. Nowlan[2]
Nabin Tiwari[1]
Jana Iyengar[1]
Syed Obaid Amin[1,2]
Bryan Ford[2]

[1]Franklin & Marshall College

[2]Yale University

Project webpage: http://dedis.cs.yale.edu/2009/tng

Once upon a time, long long ago

- TCP was the Internet workhorse
  - reliable, ordered, connection-oriented, bytestream
  - flow control (receiver throttle)

- UDP was a transport NOOP
  - Ok … it demuxed. Big Deal.

- Applications were largely happy
  - TCP generally sufficed (telnet, FTP, Email …)
  - UDP was used for simple messaging (DNS, TFTP)

Over the next several moons

- TCP continued to mature
  - end-to-end congestion control (network throttle)
  - ECN (and AQM)
  - NEW!!   MPTCP for multiple net interfaces !!
- UDP remained a NOOP
- Modern apps found services insufficient
  - realtime audio / video communication
  - multimedia streaming
  - web

New transports built in response ...

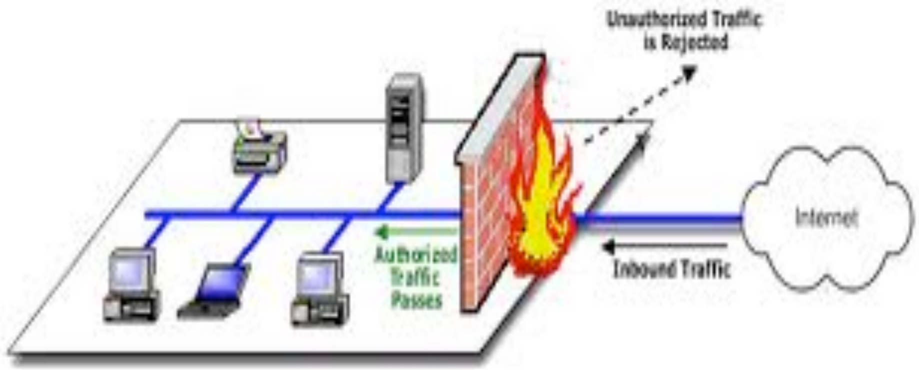
- SCTP (RFC 4960)
  - multistreaming, message boundaries, multihoming, partial reliability, congestion control
- DCCP (RFC 4340)
  - Unreliable, congestion-controlled
- SST, POC
- BXXP?

… but the Internet remained loyal!

- TCP and/or UDP get through most middleboxes
  - Only TCP gets through *all* middleboxes
  - ...often only to port 80 (HTTP) or port 443 (HTTPS)!

- New & unknown transports *rarely* get through
  - SCTP and DCCP not supported by middleboxes
  - Make it almost impossible to deploy new transports

How deep does this loyalty run?



- Network Address Translators (NATs)
  - Cheap and ubiquitous, entrenched in the network
- Firewalls
  - Rules based on TCP/UDP port numbers; often DPI
- Performance Enhancing Proxies (PEPs)
  - Transparently improve TCP (*not* UDP!) performance

Applications, in the meanwhile ...

- Build their own abstractions atop TCP and UDP
  - multiple TCP connections for multistreaming, congestion control and retransmissions on UDP
- Abstracting on UDP
  - eventually tends towards TCP over UDP
  - can interact poorly with UDP's service model
- Abstracting on TCP
  - adds buffering and latency
  - can interact poorly with TCP's mechanisms

What have we done so far?

- "NATs are evil. We won't care about them."
- "It will all change with IPv6." ← **Denial**
- "Don't design around middleboxes, that will only encourage them!" ← **Anger**
- "Wait, wait... we'll accept middleboxes, but *we'll* specify how they *ought* to behave!" ← **Bargaining**
- "Why build a new transport?? It won't get deployed anyways. Overlay." ← **Depression**

The final stage*: Acceptance

- Design assumptions for new end-to-end services:
  - Middleboxes are here to stay
  - Design should not *require* changes to middleboxes

- Consequence:
  - New end-to-end services must use protocols that appear as legacy protocols on the wire.

**Kübler-Ross model: Five stages of grief*

The Minion Suite

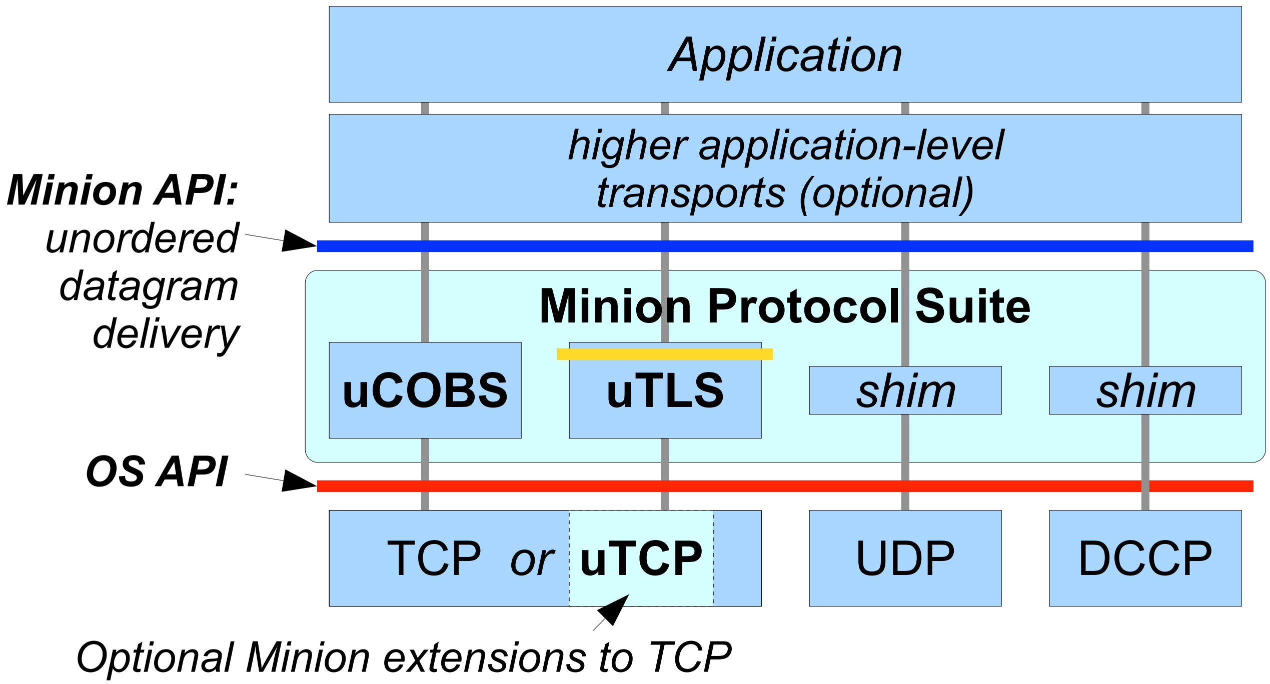
A “packet packhorse” for deploying new transports

- ***Uses legacy protocols …***
  - TCP, TLS, UDP
- ***… as a substrate...***
  - turn legacy protocols into *minions* offering unordered datagram service
- ***… for building new services that apps want***
  - multistreaming, message boundaries, unordered delivery, app-defined congestion control
  - *(may be extended to: stream-level receiver-side flow control, multipath, partial reliability)*

Outline

- Minion: a packet packhorse for new transports
  - Carry new transport services over Internet's rough terrain
- *u*COBS: unordered delivery in TCP
  - Making datagram service look like a TCP stream
- *u*TLS: unordered delivery in SSL/TLS
  - Making datagrams *indistinguishable* from HTTPS
- Impact on “real applications”

What's in the Minion Suite?



- Break up the functions of the legacy transport layer
  - "Breaking Up the Transport Logjam", HotNets '08
- Use legacy protocols as compatible building blocks
- We'll focus here on *uCOBS/uTCP* (and summarize *uTLS*)

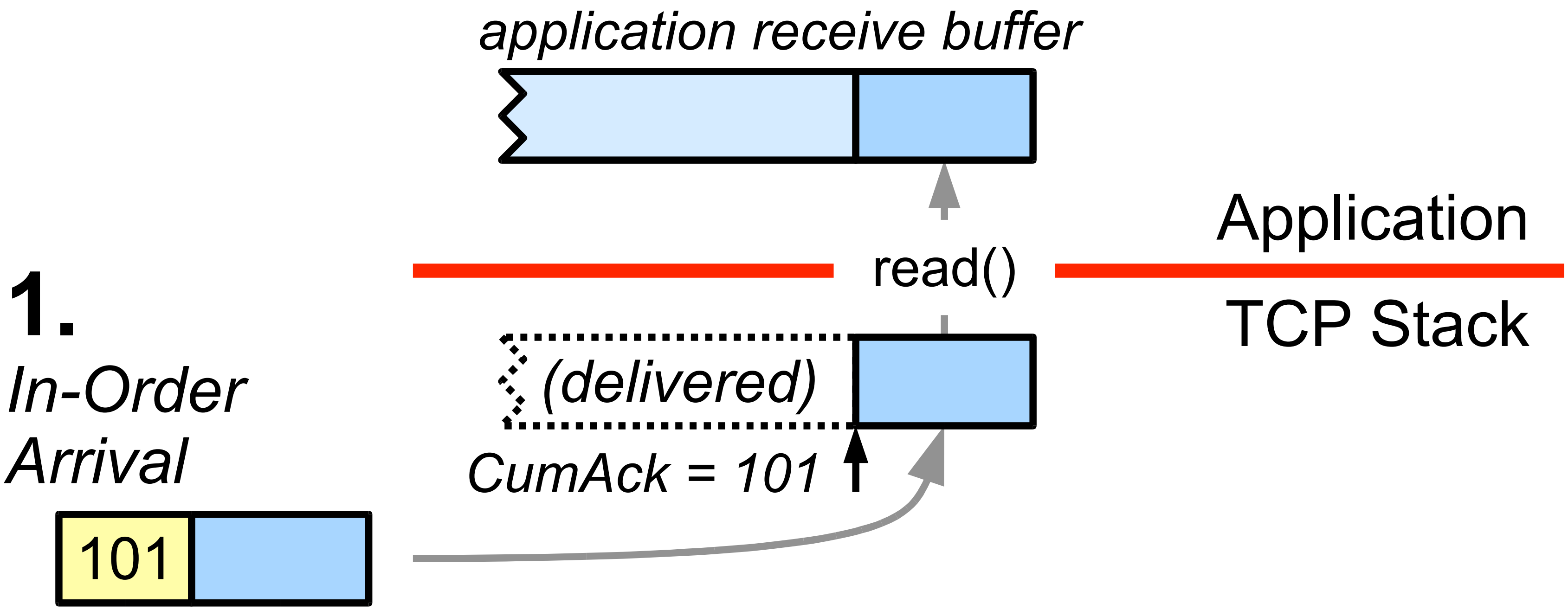
*u*TCP (*unordered* TCP)

We introduce 2 new TCP socket options in Linux:

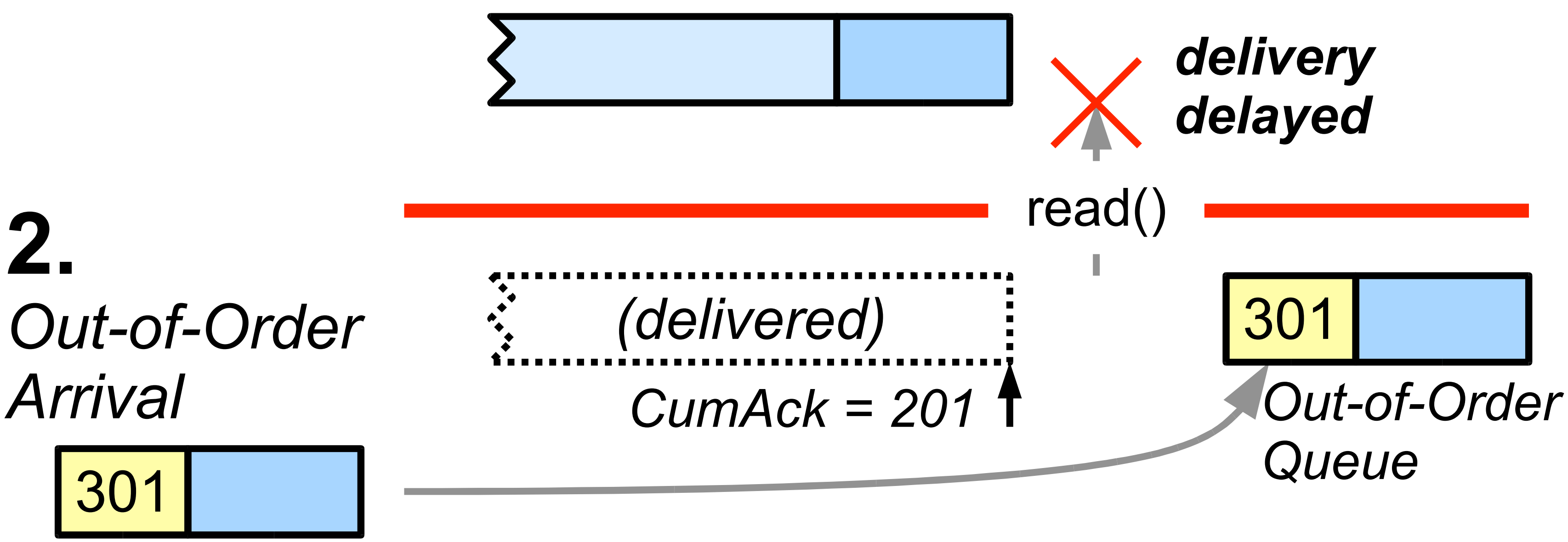
- **SO_UNORDERED_RCV**
  - kernel delivers incoming data immd
  - both in-order *and* out-of-order data
  - also delivers TCP sequence number (- ISN) with data

- **SO_UNORDERED_SND:**
  - Userspace library specifies priority with every *write()* call
  - *Message* placed in a priority queue in socket sendbuffer
  - Untransmitted data only! Transmitted data in linear queue

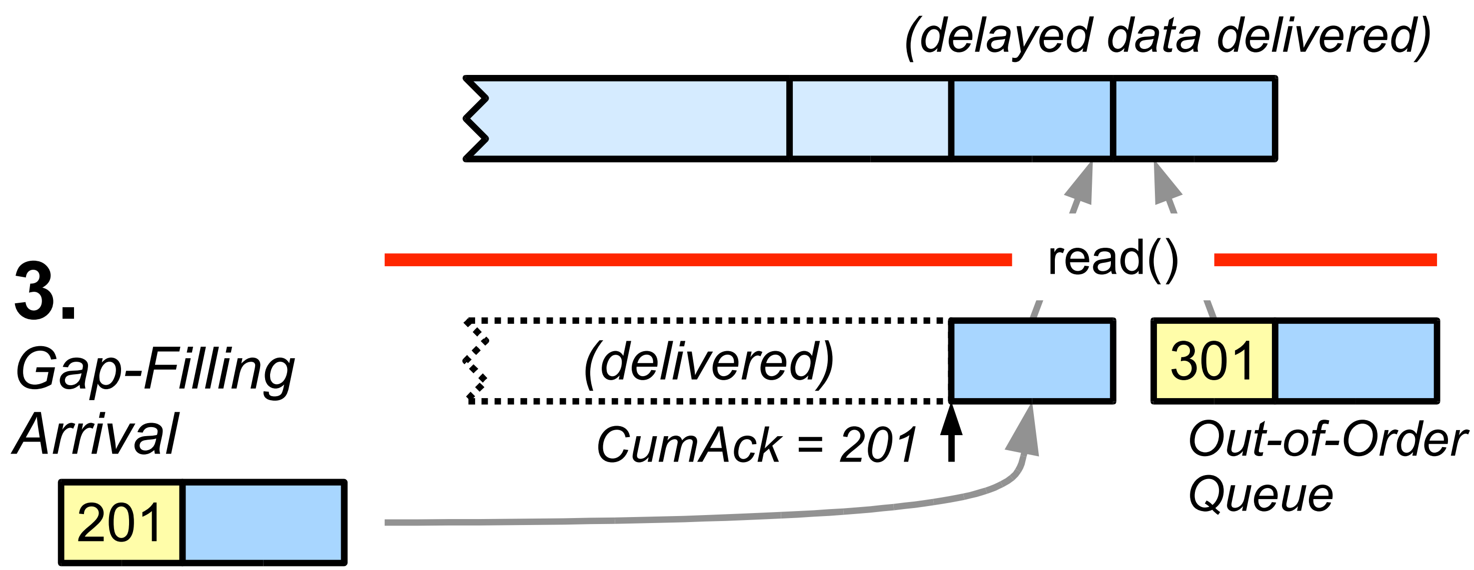
Delivery in Standard TCP



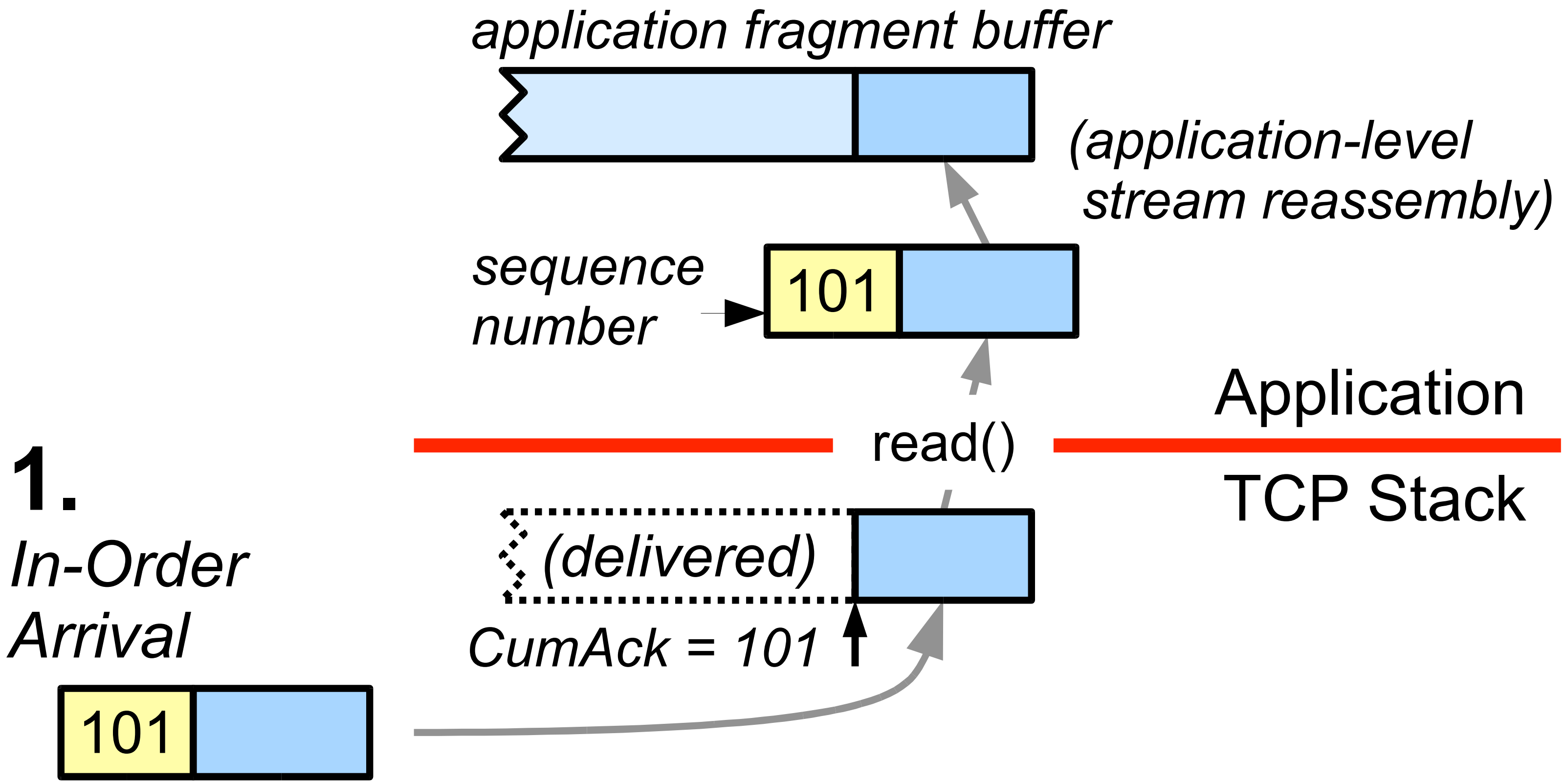
Delivery in Standard TCP



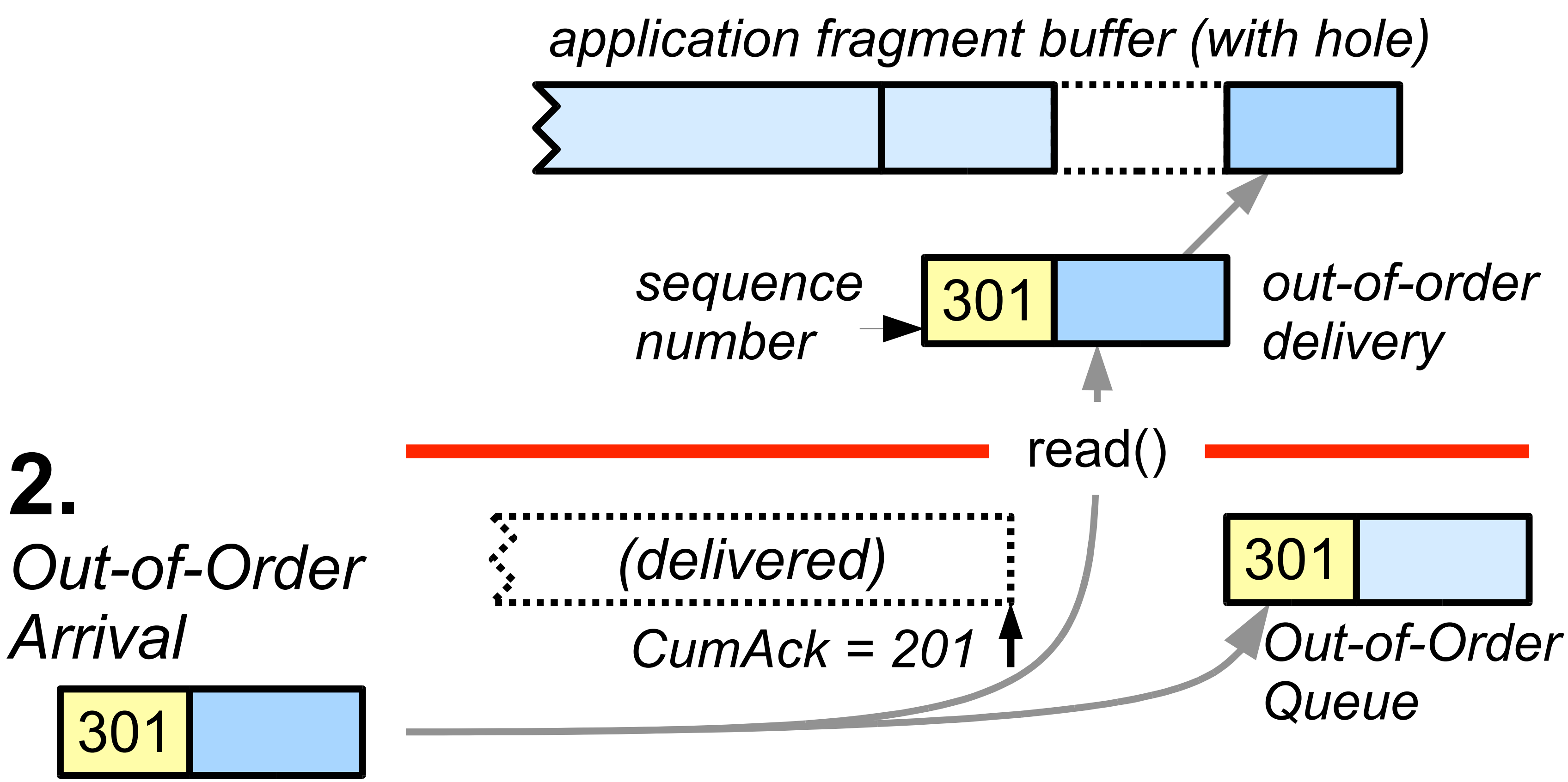
Delivery in Standard TCP



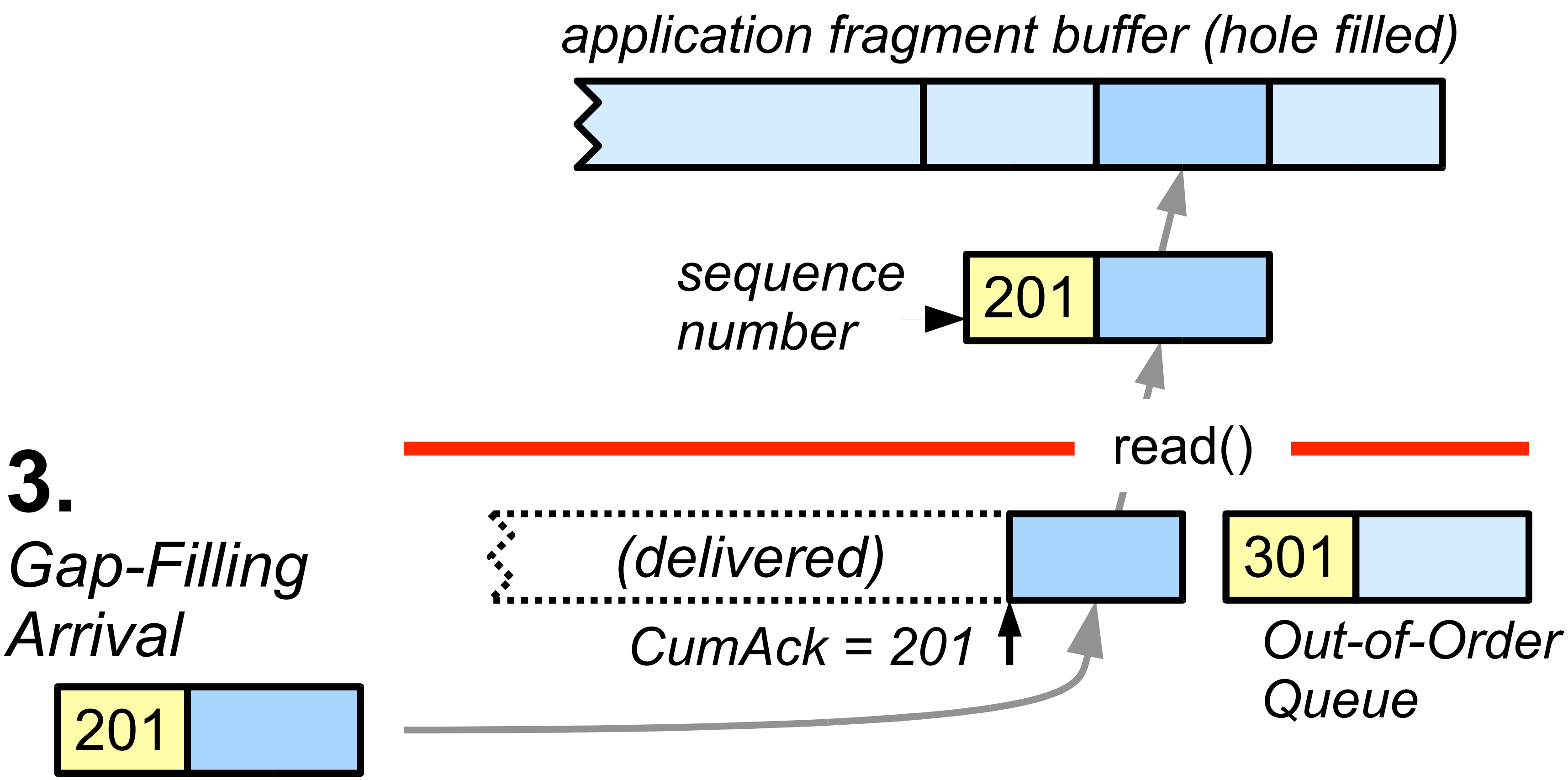
Delivery in *u*TCP



Delivery in *u*TCP



Delivery in *u*TCP



*u*COBS: Simple Datagrams on *u*TCP

- **Bytestream has no inherent structure**
  - middleboxes can re-segment TCP segments
  - need a message framing mechanism …
  - … to detect msgs in arbitrary stream fragments

- ***Self-delimiting* framing with COBS**
  - *zero* added to both ends of an app message
  - COBS encoding eliminates zeros in orig data
  - guaranteed max bit-overhead: 0.4%
    (6 bytes for 1448-byte msg)

*u*COBS: Simple Datagrams on *u*TCP

_u_COBS Sender

- COBS-encoded messages sent through *u*TCP
- with app-specified priority

_u_COBS Receiver

- manages out-of-order data received from *u*TCP
- extracts, decodes, delivers messages anywhere in received data bytes

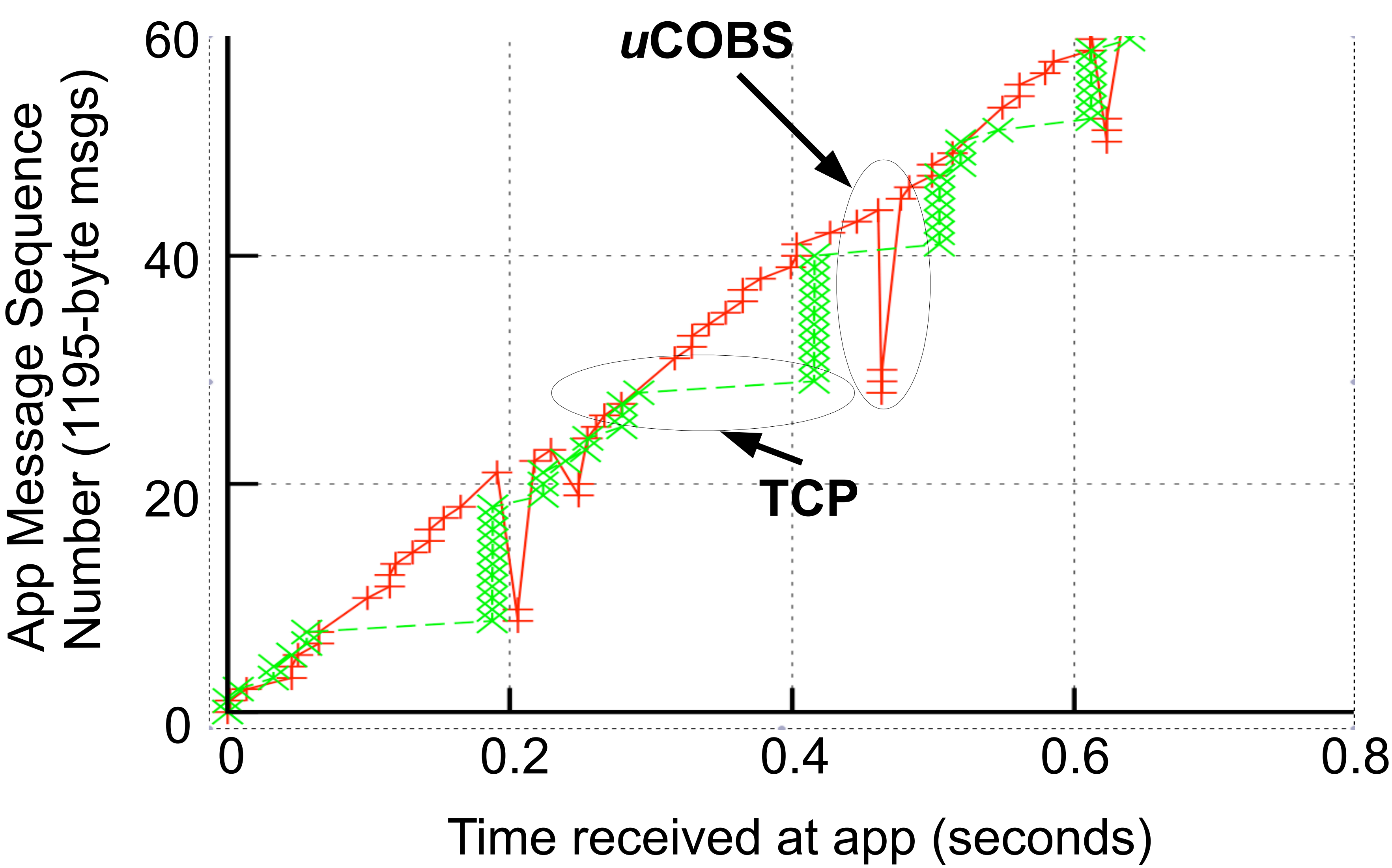
*u*TLS (Summary)

- ***u*TLS protects end-to-end signaling and data**
  - appears as SSL/TLS on the wire, *but*
  - provides out-of-order datagram service
- **Makes stream indistinguishable from, e.g., HTTPS**
  - even to middleboxes that inspect *all* app payloads!
  - only *encrypted* content affected
- **Technical Challenges:**
  - TLS records not encoded for out-of-order decoding
  - Ciphersuites chain encryption state across records
  - MACs use implicit record counter, hard to recover

Minion Implementation

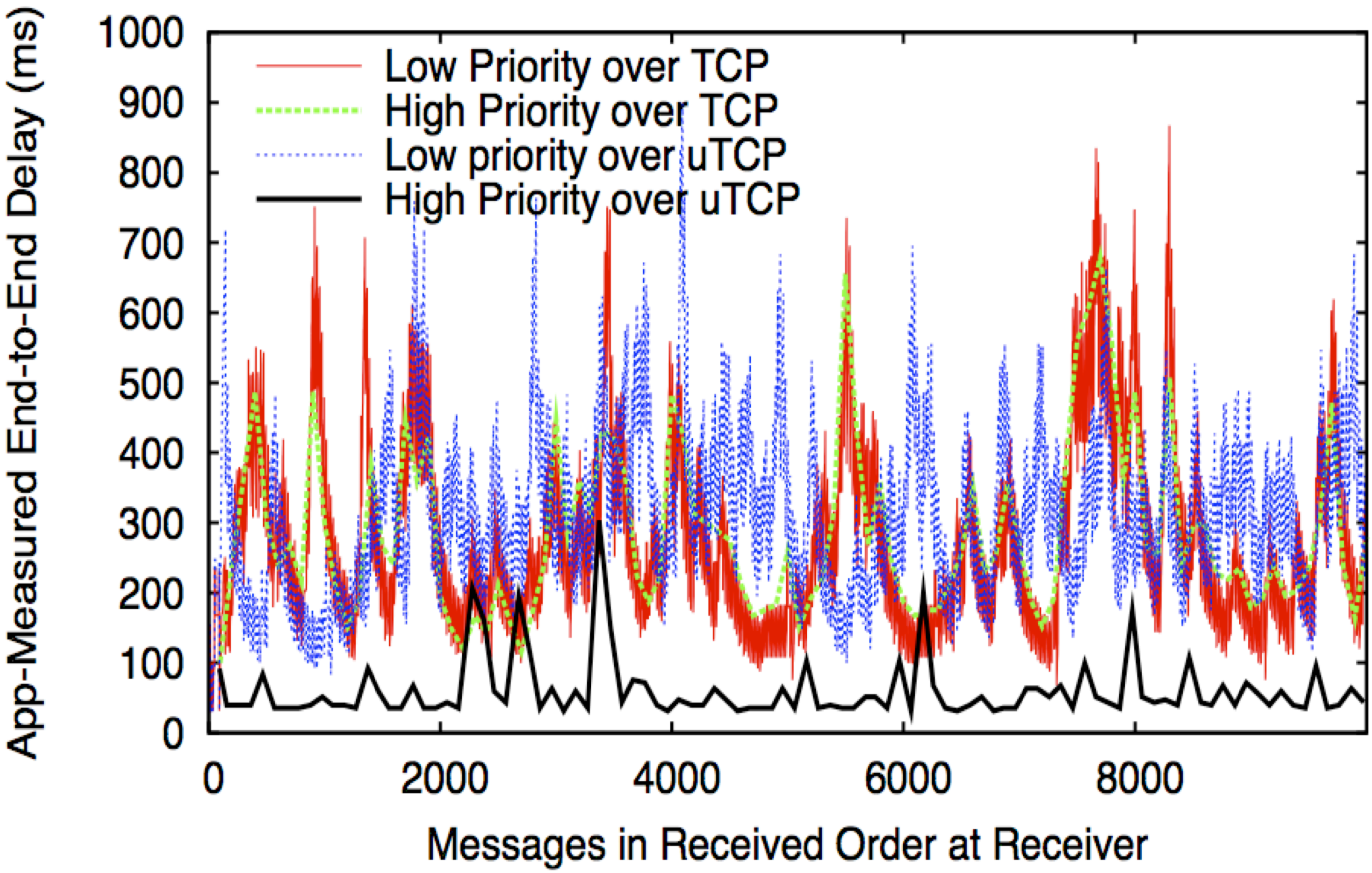
- ***u*TCP in Linux 2.6.32 kernel**
  - Added socket options to SOCK_STREAM: *SO_UNORDERED_SND, SO_UNORDERED_RCV*
  - Modified 565 (4.6%) lines of code
- **Userspace library for rest of *u*COBS and *u*TLS**
  - reassembles fragmented streams, extracts message, decodes, and delivers to app
  - library → can ship as part of apps
  - *u*COBS: 732 lines of code
  - *u*TLS: in OpenSSL, 586 (1.9%) lines of code modified

App messages with TCP (TLV encoding) vs. *u*COBS



App with message priorities

(every 100th message is high priority; 60ms RTT; 0.5% loss)



Why build Minion?

- **Instant Karma:**
  - Interactive streaming, Video Conferencing
  - Better Web browing (parallel HTTP requests)
  - Minion tunnels instead of TCP tunnels (SSL VPNs)

- **Medium-term Karma:**
  - Minion's services available at design time for new apps

- **Reincarnative Karma (if you believe in it):**
  - Next-gen transport abstraction
  - New Internet transports built and deployed on Minion

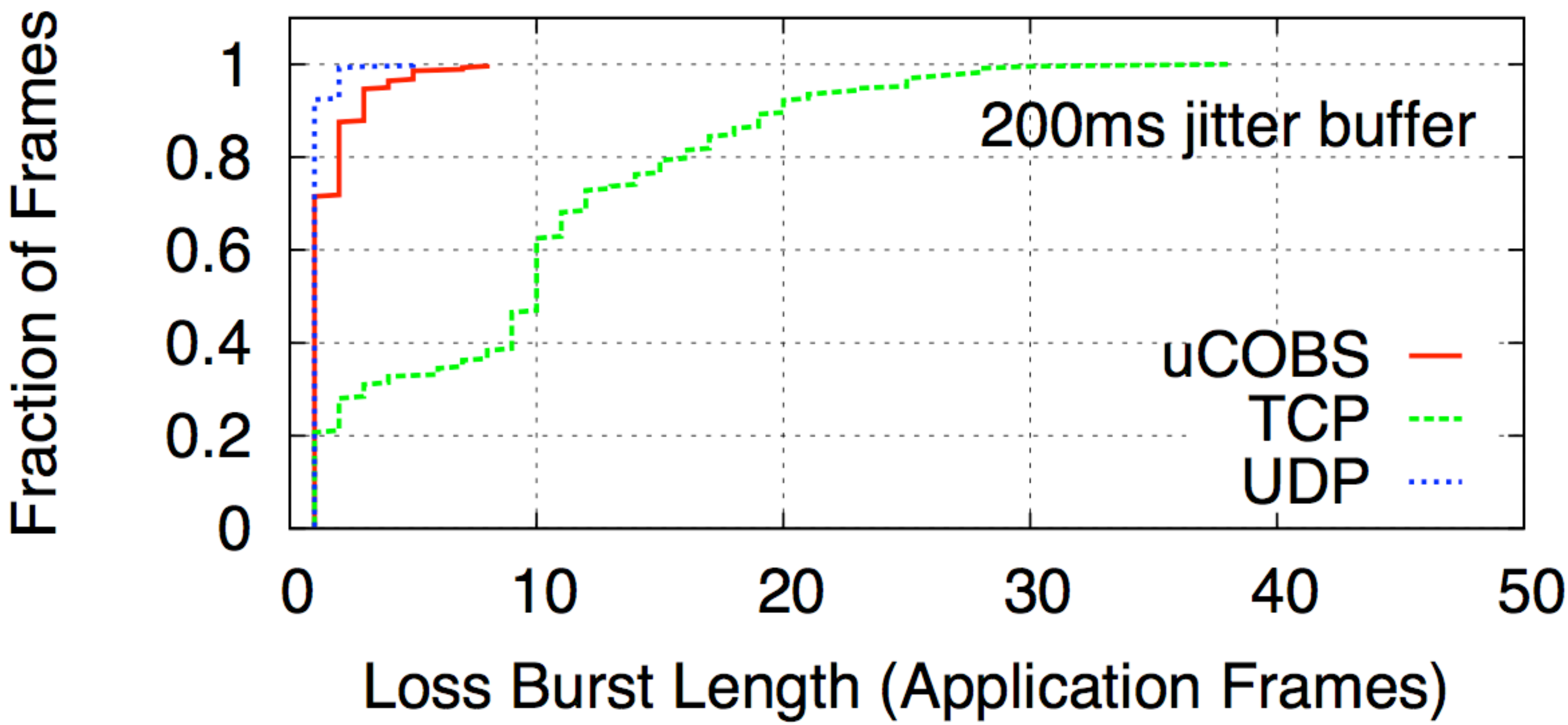
Impact on “Real Applications”

Example: Voice-over-IP (VoIP)

- Voice/videoconferencing is delay-sensitive
  - Long round-trip delays perceptible, frustrate users
- Modern VoIP codecs tolerate *individual* losses
  - Interpolate over 1 or 2 lost packets
- But are highly sensitive to *burst* losses
  - Can't interpolate when many packets lost/delayed!

VoIP application: observed delay

(3Mbps bandwidth, 60ms RTT; 4 TCP flows in background)



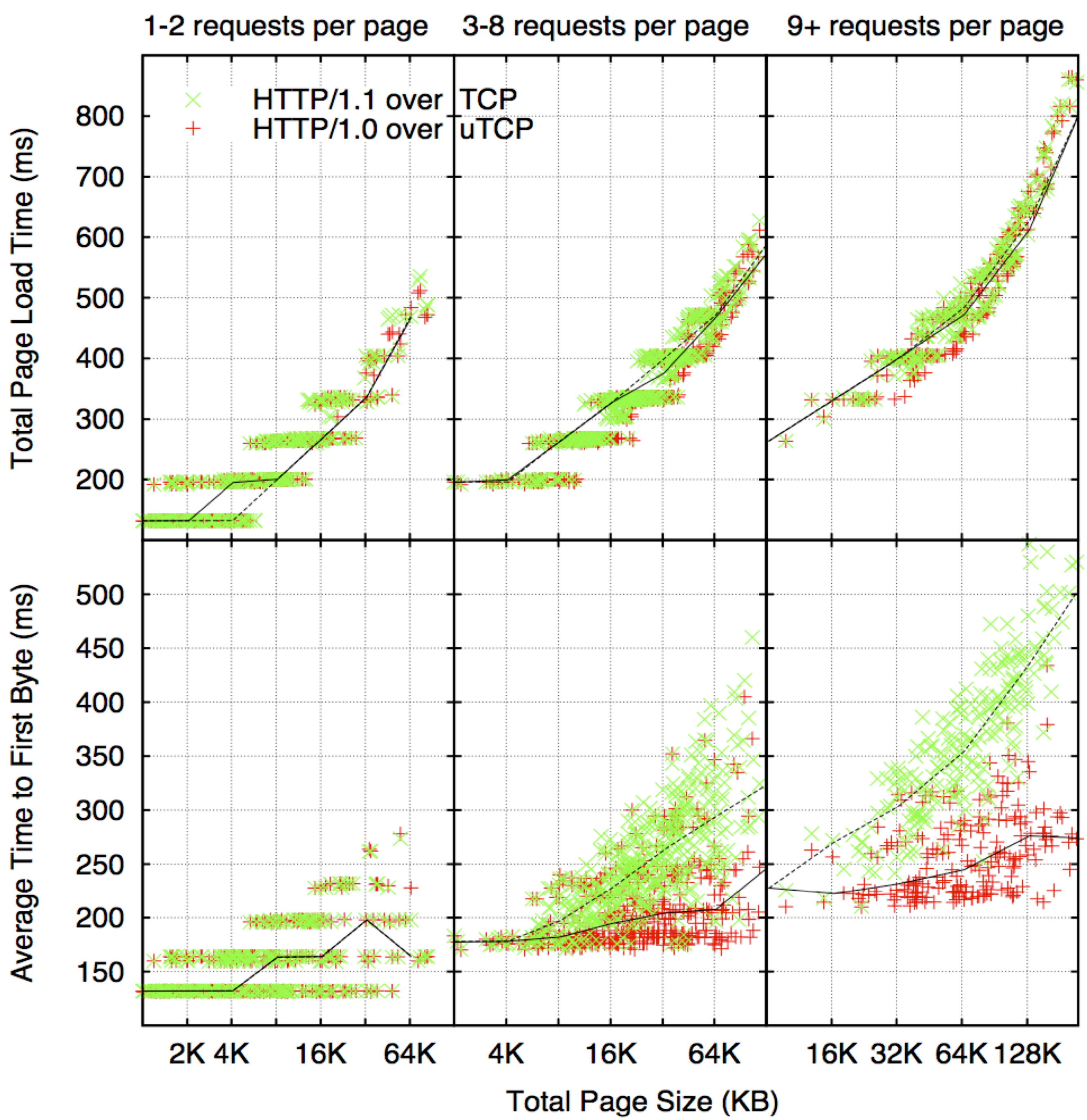
Impact on “Real Applications”

Example: Web

- Independent objects in web pages
- TCP: parallelism vs. throughput tradeoff

- Multistreaming with Minion
  - ordered streams on top of *u*COBS, 1 per object
  - sender breaks data into chunks, adds stream header, sends over *u*COBS
  - no HoL blocking at receiver across streams

Web Browsing



Trace-driven, over a network path with 1.5Mbps capacity and 60ms RTT

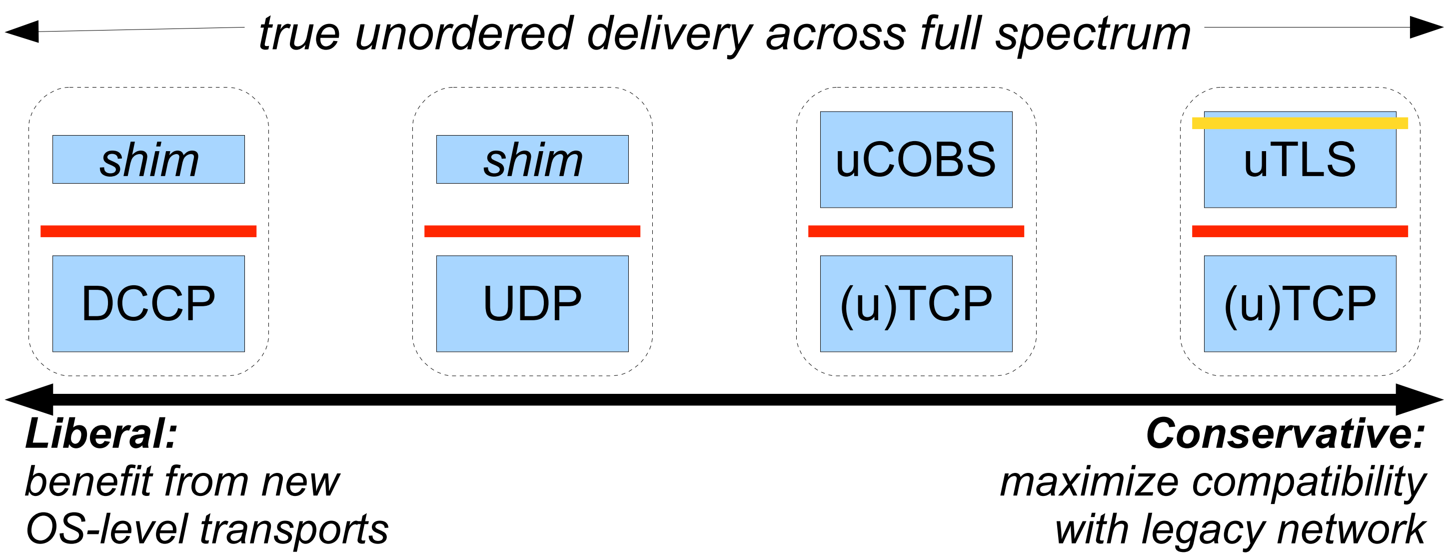
In Conclusion

- **TCP, TLS work on the Internet**
  - *workhorses* of the Internet
  - increasingly being used as substrates

- **“It's the latency, stupid”**
  - Stuart Cheshire, May 1996

- **We can fit square pegs (packets) through a round pipe (TCP, TLS)**
  - eliminates in-order delivery delays
  - most mods deployable with apps
  - turn workhorses into *packhorses*!

Continuum of configuration tradeoffs



Minion encourages adoption of new transports

- Minion allows new *services* to be created and deployed in a legacy environment.
  - Does not prevent native deployment of new protocols.
  - Encourages adoption of new protocols by middleboxes and OSes through use of new services by apps *before* middlebox/OS support is available.
- WIP: Ends need to detect *protocol-graph* supported by endpoints *and by middleboxes*
  - Negotiation Service (HotNets '09)
  - “Happy Eyeballs” on steroids

App-Observed Delay Distribution