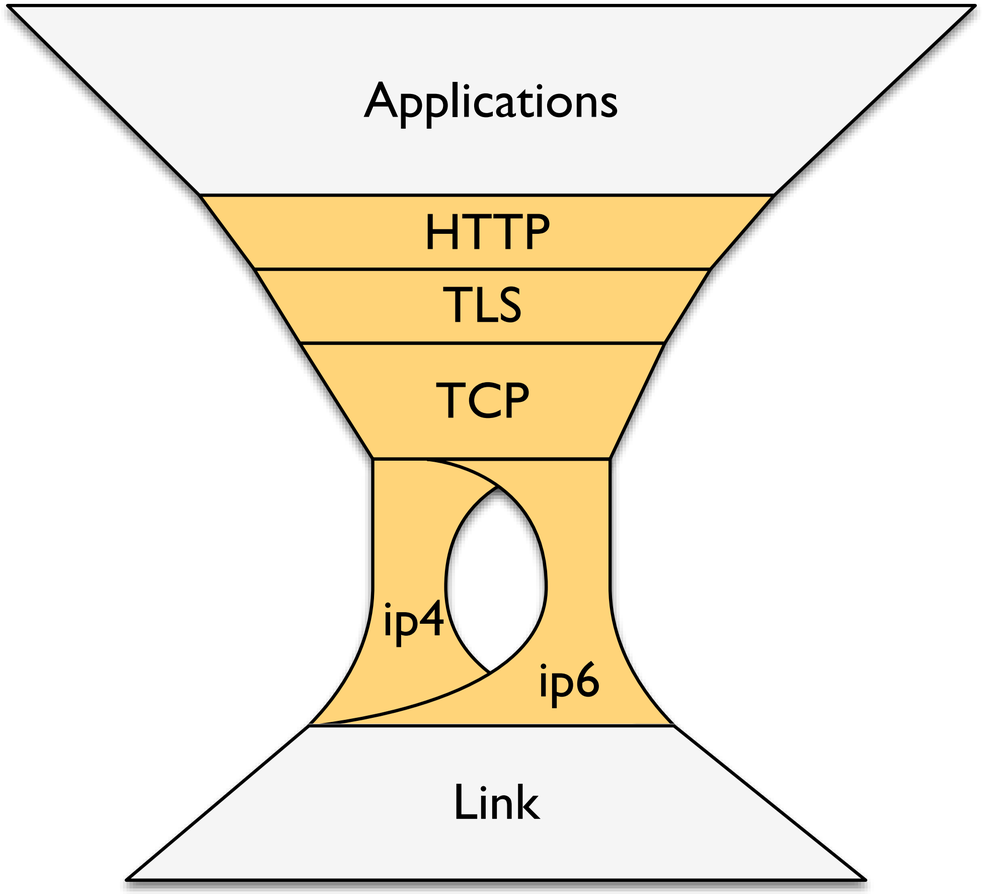

Welcome to Zürich

How did we get here?

- This is not the first room full of Internet geeks to get together to talk about the end of end-to-end...
  - (though it might be the first one in Zürich this year)
  - Deering "Watching the Waist of the Protocol Hourglass", IETF 51 (2001)
- Theme is of ongoing concern to the Internet Architecture Board
  - RFC 5218 What makes for a successful protocol? (2008)
  - RFC 7305 / IAB Workshop on Internet Technology Adoption and Transition (2013)
- At the time of Deering's talk, NAT was identified as a key challenge to the architecture.
- Since then: the boxes in the network are getting smarter and smarter
  - Economics: Moore's Law, value-add for operators, scalability in enterprises, something vendors can vend.
- But not smart enough to preserve end-to-end extensibility in transport.

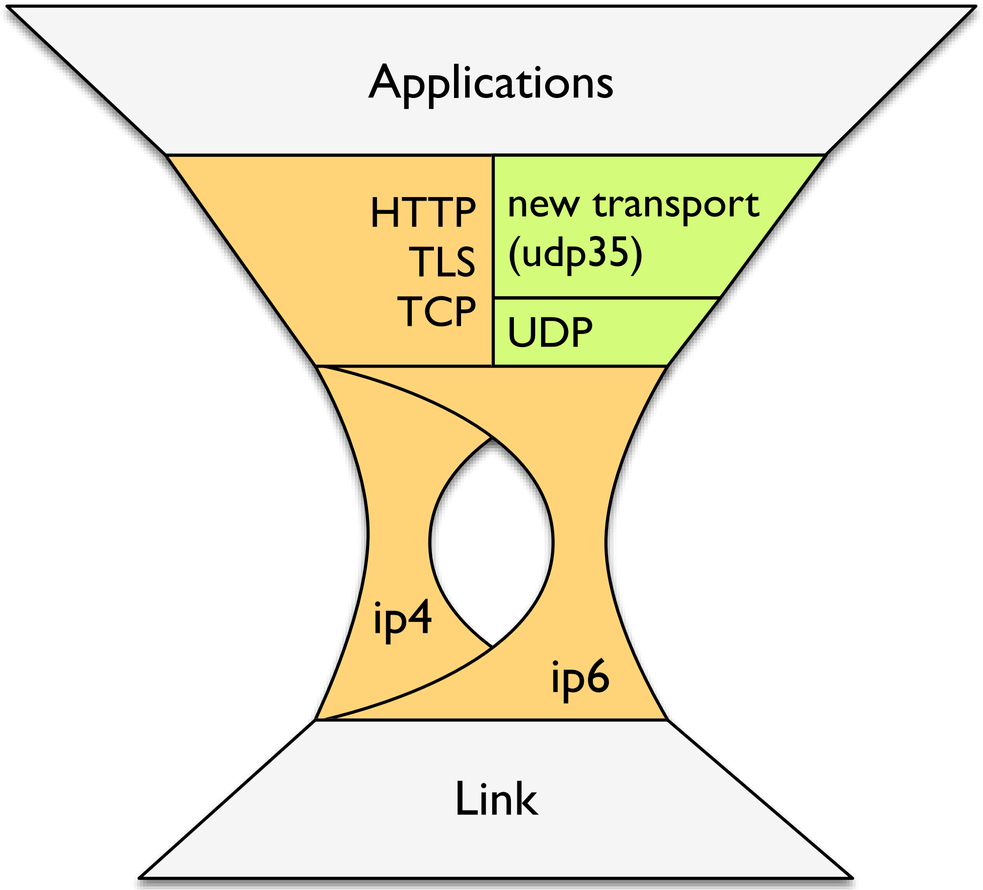
In the meantime...

- The interface at the endpoint is largely the same as it has been:
  - "The network is a file descriptor"
- The waist of the hourglass has crept up to HTTP: even less flexible.
- Transport is squeezed in the middle.
  - Way out: applications implement their own new transport features.



IAB Stack Evolution Program

- How to support evolution of the transport given today's interfaces and today's Internet?
- Initial suggestion: foster development of new transports over UDP, shim layer for middlebox cooperation.
- Next step: hold a workshop to make sure we've defined the problem correctly and can pull off pieces of it which are actually solvable.



Why now?

- Two reasons we think now is the right time to have this discussion:

1. new energy in the IETF:
   - work which requires flexibility we don't think we have (e.g. WEBRTC, TCPINC)
   - work to provide that flexibility at the interface (e.g. TAPS)
2. pressure created by increasing deployment of encryption:
   - "Everything over TLS" effectively turns many middleboxes into expensive heaters.
   - Opportunity to strike a balance between endpoint and midpoint requirements.

The Bigger Picture

What is this about?

Some viewpoints expressed in position papers:

- Security vs. middlebox tussle
  - Many stakeholders depend on DPI to implement their policies and services
  - Users increasingly desire privacy that makes DPI hard or impossible
- Perceived need for broader range of/flexibility in transport services
  - taps WG – recognizing benefits of factoring functions out of applications and doing them once
- The rise of “path” (as distinct from “route”) as an important network element
  - NFV, differentiation of application treatments

The Opportunity

- Incentives are aligning
- Recognized need for evolution/innovation
  - This situation may not come along again for a long time!
- Challenge: Scope
  - Narrow: Solve immediate problem(s) only.
    - Focus on engineering; Accept partial solutions
    - Disturb the status quo as little as possible
  - Broad: Take architectural view, tackle general problem
    - Recognize that tussles ⇒ missing mechanisms
    - Aim for (eventually) a solution that works for the next 30 years

Separable Concerns

Components of a solution:

- A framework for composing end-to-end functions (features?)
  - Wire format
  - Basic semantics
  - Initial set of functions: demux, framing, security, ...
- A mechanism (protocol) for determining/ negotiating path characteristics
  - End-middle-end
  - In-band vs. out-of-band?

Desiderata for E2E Feature Composition Framework

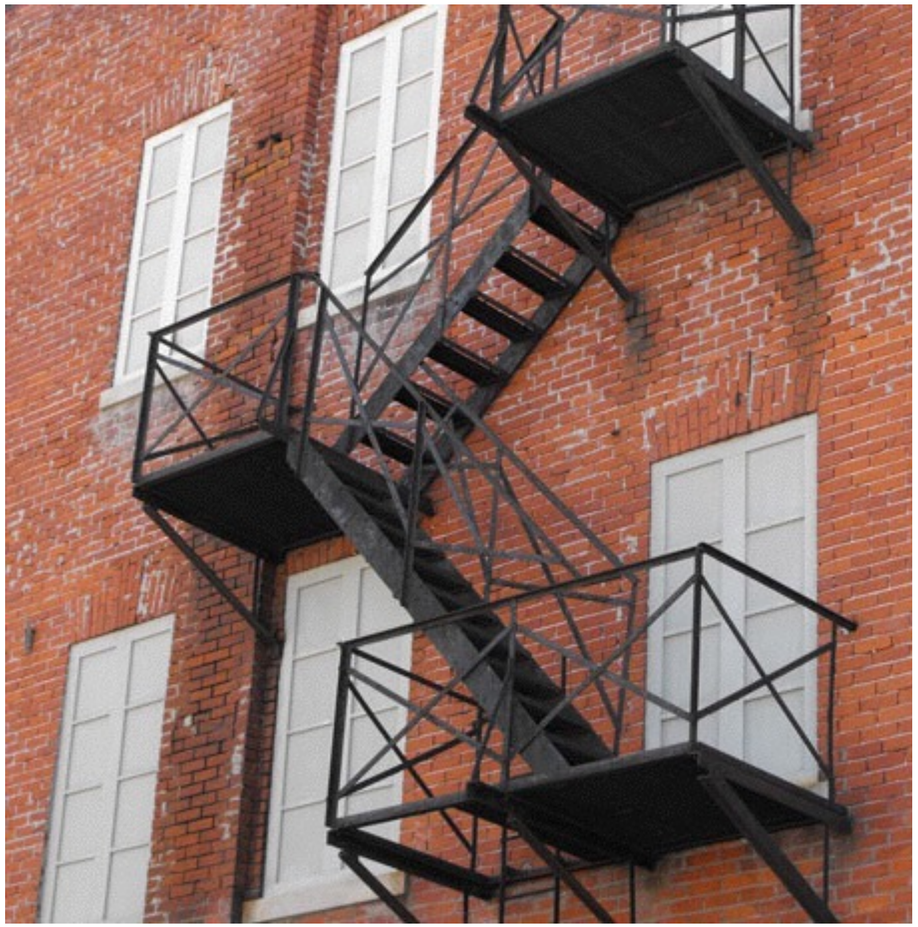
- Fine-grained control over functionality and visibility
  - Both header and payload
- Integrate security
  - Including negotiation of other functionality
- Admit trusted intermediaries
- Do not constrain functions to be processed serially (except where required)
- Allow evolution both above and below
  - Don't assume demux is provided below
    Even if initial encapsulation is in UDP
  - Support recursive application
- Support endpoint migration

The Secure Transport Tussle

Steve Dunleavy, *Creative Commons Attribution 2.0 License*

- Unilateral deployment of boxes
  NAT, firewalls, load balancers, intrusion detection systems, transport relays, content caches…
- Versus end-to-end transparency
  Forward error correction, multiple streams, encryption, authentication, multi-path, mobility…
- So far, the boxes are winning
  Could not even deploy ECN!

Precedent: the UNSAF NAT Traversal



- Partial success only for being nice
  UPNP IGD, NAT-PMP, now PCP
- Unilateral hole-punching worked
  STUN, Teredo based on observed behavior of NAT
- Tussle has now settled
  Home routers don’t want to break Skype, Xbox
  TURN as cost of doing business
- The IETF did not help much
  Long story…

UDP + Encryption ➔Secure Transport

IP + UDP header	Context identifier + sequence number	AEAD checksum	Encrypted content, including transport headers

Figure 1 QUIC packet format

- Can be deployed today, wherever UDP is supported
  - HTTP/HTTPS fallback when UDP is blocked
  - Require UDP load-balancer for big web sites
  - Google doing it with QUIC
- Resets the Transport vs. Middlebox tussle
  - Boxes, e.g. load balancers, need to explicitly cooperate with the endpoints
- Encryption is essential!

Winning the Secure Transport Tussle

- UDP + Encryption, e.g. QUIC
  Not just for the web…
  Don’t put the IETF in the way
- Observe better latency
- Get friendly middle-boxes
  Stateful firewalls
  Load balancers
- Get enterprise deployments

What Makes for a Successful Protocol? (RFC 5218)

Success most easily comes when the natural incentive structure is aligned with the deployment requirements.

it is best if there is significant positive net value at each organization where a change is required.

Tussles stem from conflicting incentives

- Incentive for e2e protocol innovation inherent in endpoint (app/host/server) competition
- Incentive for middlebox optimizers inherent in network operator competition
- Incentive for middlebox firewalls inherent anywhere one is worried about untrusted entities
- Incentive for middlebox inter-protocol proxies inherent in innovation

These are all natural causes in a free market

How to deal with these incentives

- Upper layer protocols:
  - Incentive is usually to “work anywhere”: must be capable of using, or looking like, HTTPS
  - Incentive is usually to “be efficient” (throughput,latency, power, etc.): must be capable of using most efficient mechanism that works
    - e.g. if session is *idle*, UDP is very expensive for power etc.
  - Incentive is usually to “be as simple as possible”: easy to diagnose, maintain, low footprint, etc.

There is a tussle even among the above, so each designer has to make tradeoffs

How to deal with these incentives

- Lower layer protocols:
  - Anything optional and not used by default will be filtered intentionally or as a side effect (e.g. of proxying), by someone
  - ***Must be (but often isn't) taken into account at protocol design time***

The Internet's Ossification

Problem 1: Middleboxes are commonplace

- They enhance the functionality of the network, so they are not going away
- Presence restricts what is considered "correct" traffic, more often than not in an opaque way


carrier-grade NAT


ad insertion


IDS


BRAS


WAN accelerator


session border controller

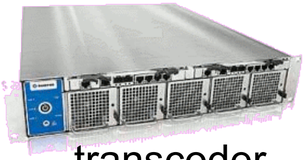

load balancer

DDoS protection


firewall

QoE monitor


DPI


transcoder

The Internet's Ossification

- Problem 2: Difficult to deploy new protocols on end systems
  - Even with middleboxes, still possible to deploy extensions to protocols (e.g., a recent study says 86% of Internet paths allow TCP extensions)
  - New protocols often require changes to kernels → long deployment cycles

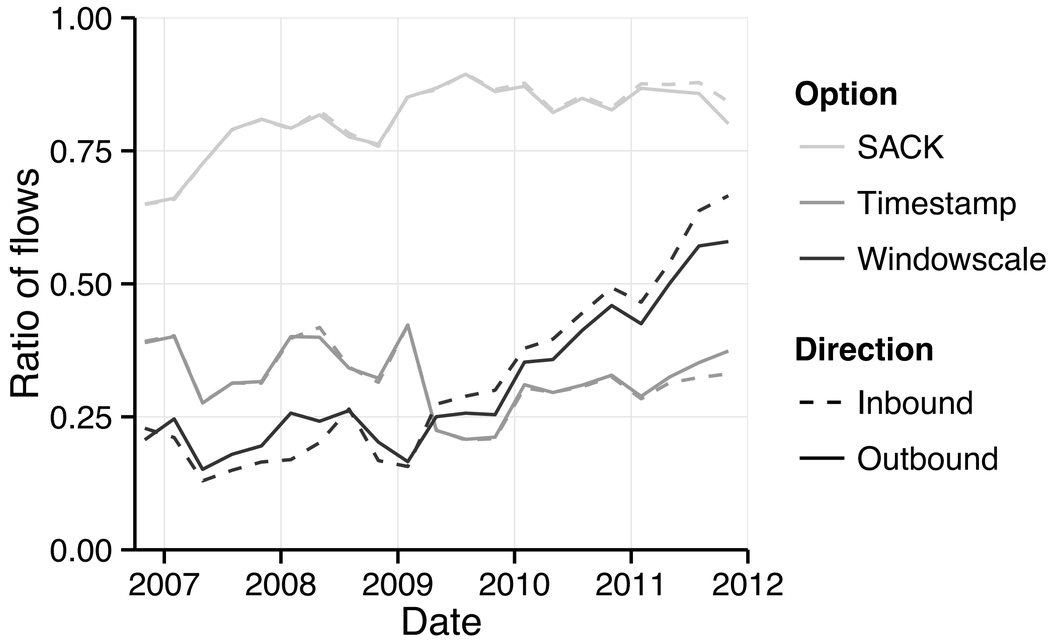
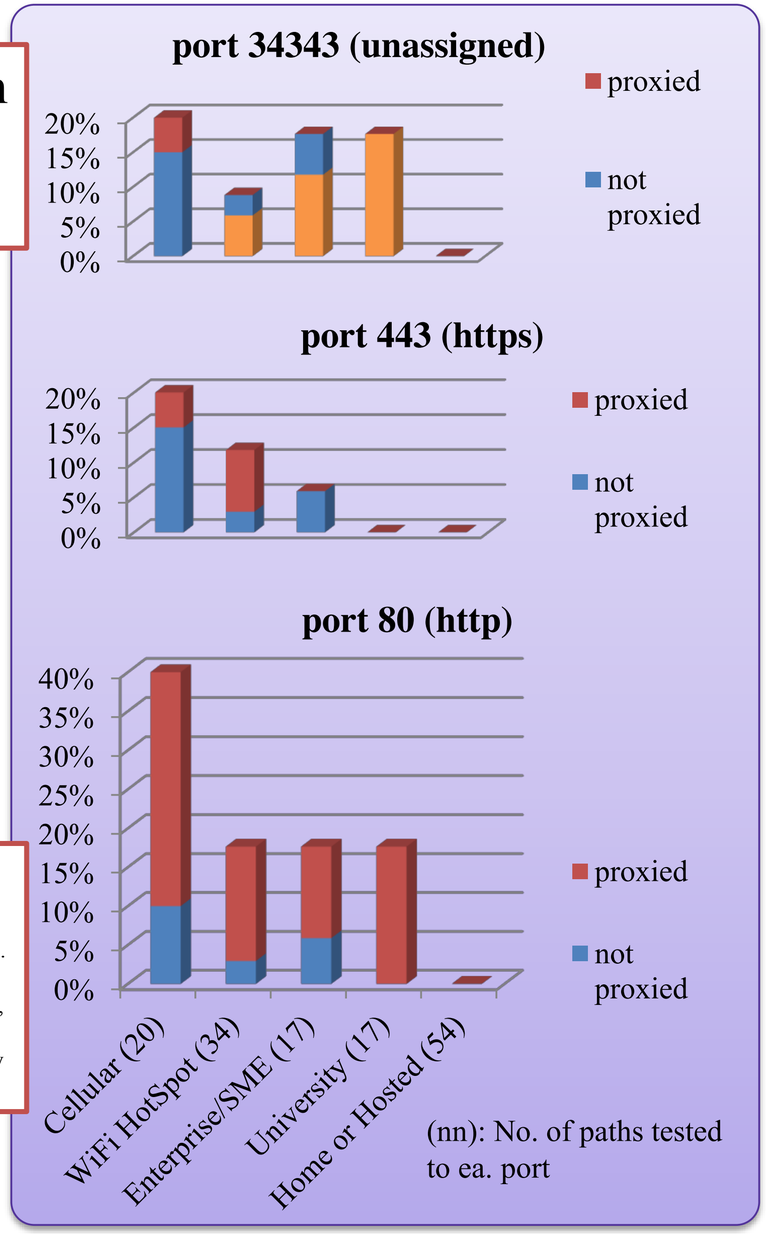


Figure: TCP options deployment over time.

the old transport extensibility architecture

Unknown option stripped from TCP SYN



Source: The dataset collected for:

Honda, M., Nishida, Y., Raiciu, C., Greenhalgh, A., Handley, M., and H. Tokuda,
"Is it Still Possible to Extend TCP?",
Proc. ACM Internet Measurement Conference (IMC'11) 181--192, Nov 2011

Overview of Solution Approaches

Middlebox cooperation

- Hardie, T.: Network Function Virtualization and Path Character
- Huici, F., Raiciu, C. and Honda, M.: In-Network Processing, User-Level Stacks and the Future of Internet Evolution
- Nádas, S. and Loreto, S: Middleboxes in Cellular Networks
- Raiciu, C., Olteanu, V., and Stoenescu, R.: Good cop, Bad Cop: Forcing Middleboxes to Cooperate

Transport layer

- Black, D.: UDP Encapsulation: Framework Considerations
- Briscoe, B.: Tunneling Through Inner Space
- Huici, F., Raiciu, C. and Honda, M.: In-Network Processing, User-Level Stacks and the Future of Internet Evolution
- Nottingham, M. and Ponec, M.: UDP-based Application Layer Protocol Recipes to the Rescue
- Raiciu, C., Olteanu, V., and Stoenescu, R.: Good cop, Bad Cop: Forcing Middleboxes to Cooperate
- Reddy, T., Patil, P., Wing, D. and Versteeg, B.: WebRTC UDP Firewall Traversal
- Schmidt, P. and Enghardt, T.: Cross-Layer Coordination: Let's Talk About Intentions
- Welzl, M., Fairhurst, G., and Ros, D.: Ossification: a result of not even trying?

The role and rule of middleboxes

Motivations

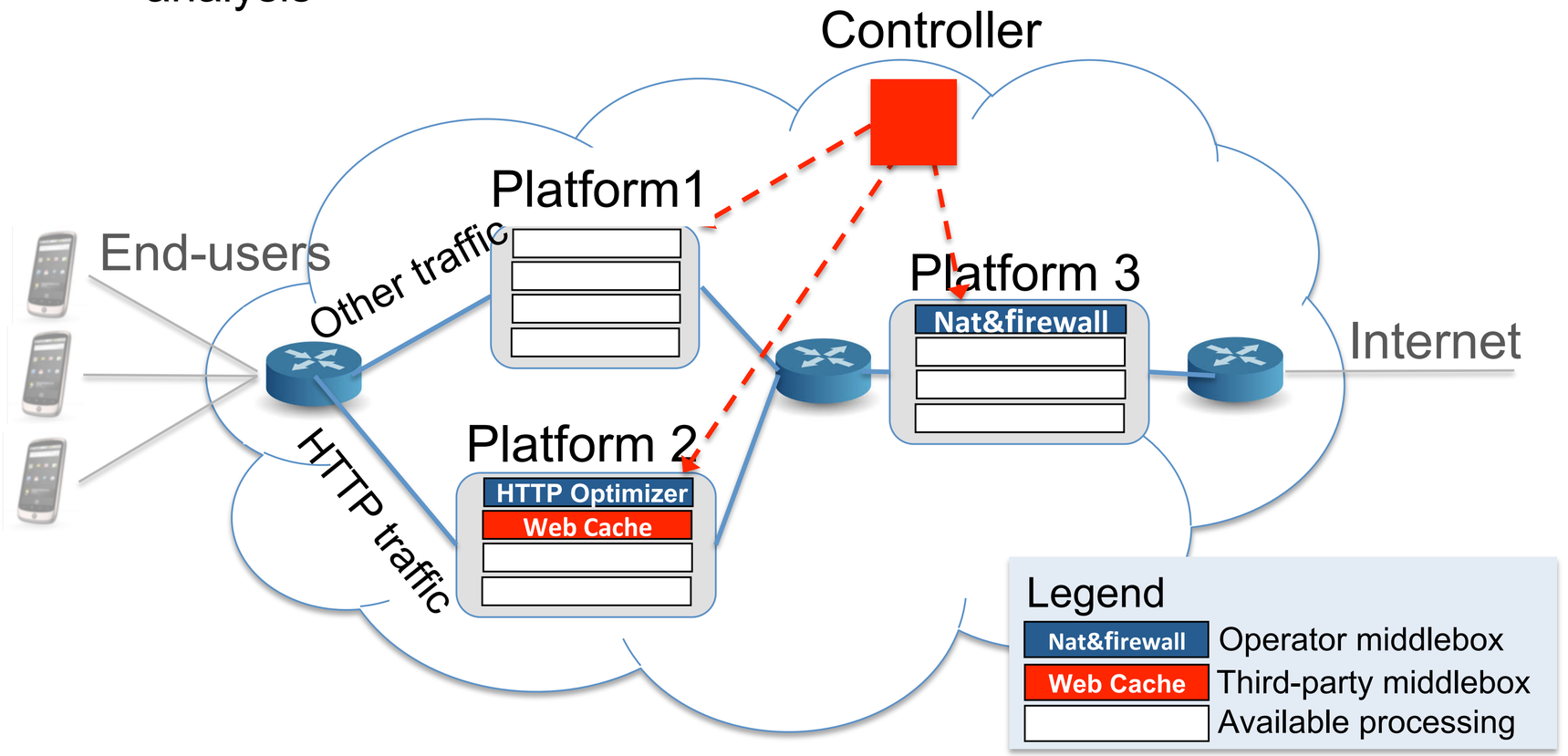
- Middleboxes are more and more prevalent in nowadays network architecture ([Sherry et al. 2012])
- Middleboxes have a negative impact on TCP evolution ([Honda et al. 2011], [Hesmans et al. 2013])
  - potentially, middleboxes may impact any new protocol
- There is a wide variety of middleboxes
  - who's doing what?
- Providing a "bestiary" of middleboxes potential interference is of the highest importance

Classification

- Path-impairment oriented middlebox policy taxonomy
  - categorizes the initial purpose of middlebox policy
  - categorizes middlebox' potential unexpected complications
- 3 main categories
  - *action*
    - ✓ the fate of a packet crossing a middlebox implementing this policy
  - *function*
    - ✓ the policy purpose
  - *complication*
    - ✓ the possible resulting path deterioration

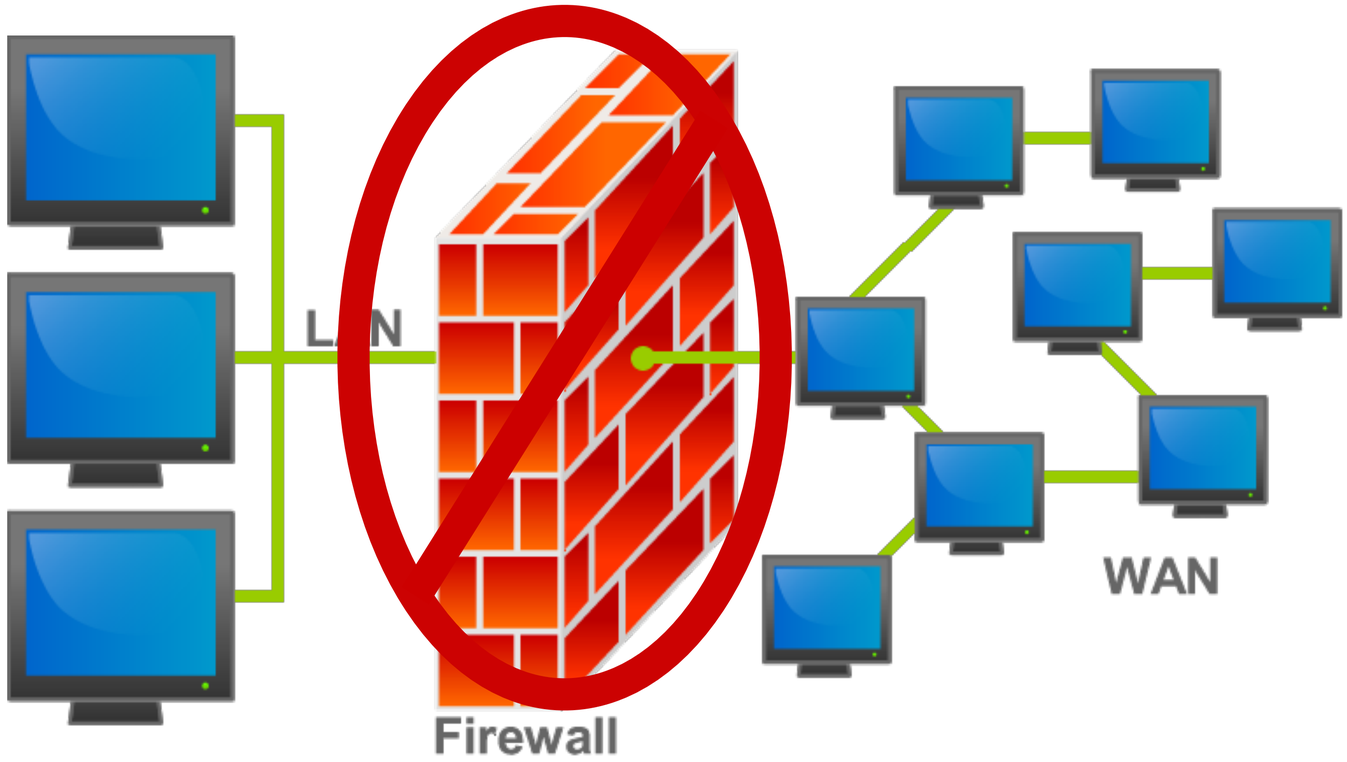
In-Network Processing

- Virtualized, software-based network processing deployed in operator networks by operators and third parties
  - Platforms can be explicitly addressed, security checked with static analysis



Good news!

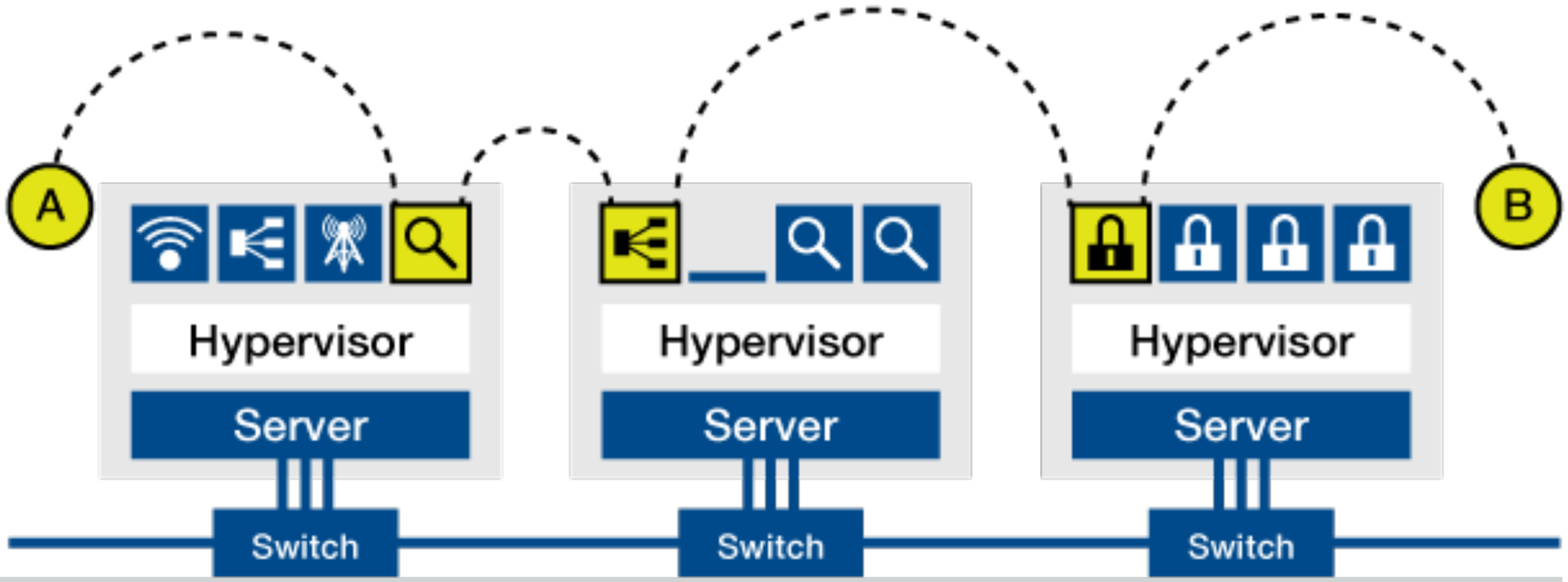
Middleboxes are going to go away!



Not so Good news!

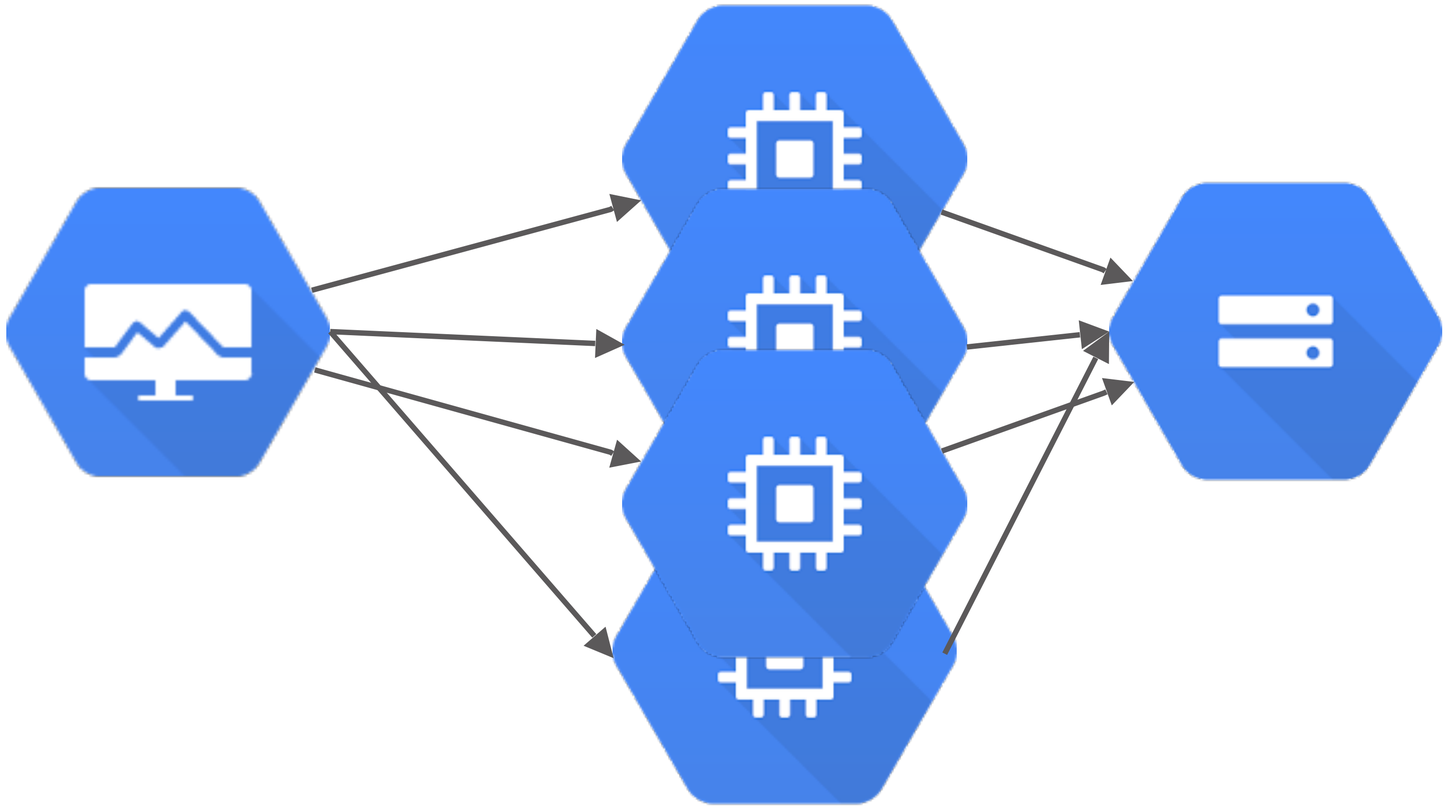
Their functions won’t.

Instead, the same network functions will be chained as virtualized services.



Service chaining may not be 1-to-1

As load increases, instances get added.



What happens to the path?

Each new instance may* add a new queue.

The network path may also change. Remember, the network is routing to services; if the next service or instance is distant, it gets looped in regardless.

This may

create bufferbloat

induce congestion later in the service chain or network

cause variability in the one-way network latency

What should we do about it?

Deploy NFV/service chaining with this in mind.

Most radically, make the VFs part of the host stack

Decide whether/when VF instances should use ECN/participate in AQM.

Account for service chain variability in interpreting path variability.

Mobile Broadband

Cellular properties

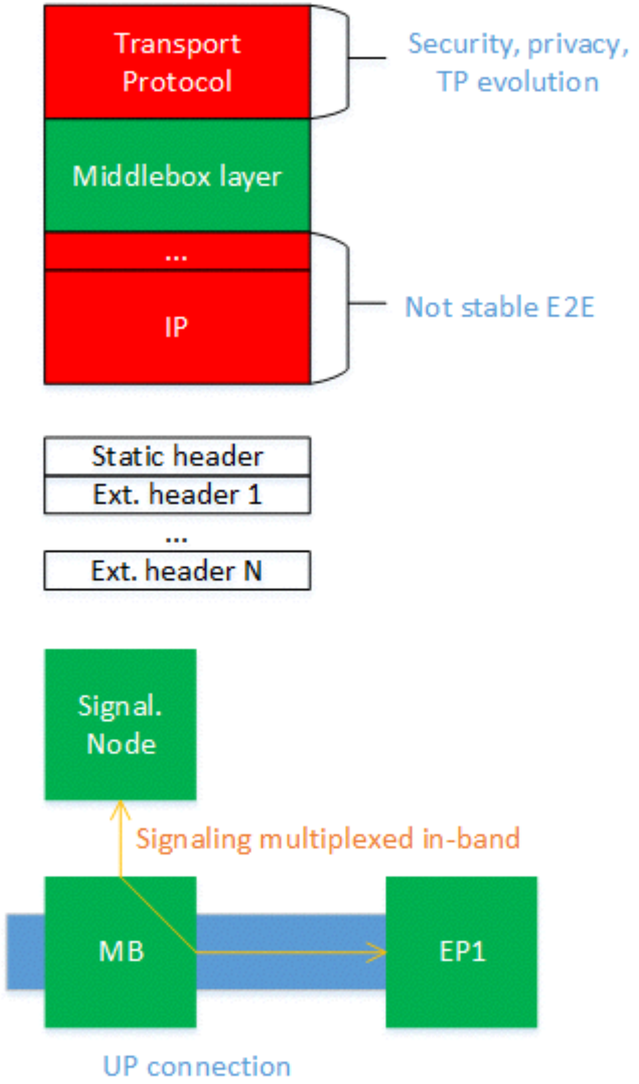
- Optimizing resource sharing among streams due to
  - Variability of congestion level
  - Cost of radio resources
- The Radio Resource Management algorithms can benefit from stream classification
- Redundancy between Transport Protocols and network functions

Related middlebox roles

- Policy Decision Point
- Providing information about network path
- Sending advisory messages to applications
- Transport Protocol performance enhancement
- Application layer optimization

Role of a Middlebox layer

- A lightweight approach shall be possible
  - To minimize privacy impact
  - Does not rule out a more versatile approach for cooperating end-hosts
- The right place looks to be between IP and the TP
  - Might be over UDP, especially in legacy
  - It allows some middlebox features to be TP independent
  - Might be used for legacy TPs
- It may have static fields and extension headers in each packet
- It may allow multiplexing a signaling protocol in-band
- It may benefit from pre-configuration when a device connects to a domain



Unpredictability leads to inefficiency

- Middleboxes make Internet unpredictable
  - Connectivity depends on protocols, port numbers and even payload
  - Packets may be changed arbitrarily in flight
- *New apps **must** act like old ones* ➔ tunneling is the norm
  - Tradeoff between efficiency and reachability
  - Most apps choose reachability

Two approaches

- **Good cop**: constructive solutions, assumes ISPs want to cooperate
- **Bad cop**: force a predictable contract (*„ninja tunneling“*)

➔ Together, these should force ISPs into cooperating with endpoints

Raiciu, C., Olteanu, V., and Stoenescu, R.: Good cop, Bad Cop: Forcing Middleboxes to Cooperate

Good cop An API to talk to Middleboxes

Clients are given an API to query the contract they will receive from the network

```
reach <node> [flow] -> <node>[flow] [const fields]
```

Examples

UDP inbound reachability on port 1500

```
reach internet udp->client dst port 1500
```

TCP reachability on port 80 and immutable payload

```
reach internet tcp->client src port 80 const payload
```

Good cop Implementing the API

- ISP deploys a controller that knows deployed middleboxes and router configurations/FIBs
- Controller uses static analysis to answer client queries
- Controller may also reconfigure middleboxes to offer desired service

Bad cop Ninja tunneling

Clients use multiple tunnels <u>simultaneously</u> to send traffic

- *Tunnels include native IP, UDP, TCP, HTTP/HTTPS, DNS, covert channels and can change dynamically*
- *Traffic spread over the different tunnels with MPTCP*
- *Drop tunnels that where packet changes are detected*

Benefits

- *Unchanged apps*
- *Reachability is ensured in all cases*

***Key difficulty**: ensuring most traffic flows over the most efficient tunnel, and MPTCP congestion control can help*

Raiciu, C., Olteanu, V., and Stoenescu, R.: Good cop, Bad Cop: Forcing Middleboxes to Cooperate

Unbreaking transport

Duelling Approaches

- "TCP is broken, burn it to the ground!"
  (new transport over UDP encapsulation)

- "Long live TCP!"
  (return flexibility and extensibility to existing transports)

Design UDP-based protocols when TCP is inadequate to meet the needs

- New protocols to replace TCP have issues with deployment/compatibility and TCP itself is evolving too slowly
- Protocols built on top of UDP provide immediate solutions for deployment today
  - Note protocol**s** not protocol, no one-size-fits-all framework/solution can help address the needs of everyone (efficiently)
- Provide "small pieces loosely joined" specs that application protocols can choose to use (at least as an inspiration)
  - Could point to existing specs and their applicability, support/advise on it long-term
  - Focus on topics/problems shared by many protocols: conn setup and reuse, fairness, congestion&flow control, reliability, integrity, signaling, authentication, encryption, extensibility, standardization, firewalls, OS support, …
- New UDP-based transports keep appearing
  - Examples: Google's QUIC, P2P protocols, Akamai's Hybrid HTTP/UDP protocol

Ossification is essential – it implies backwards compatibility

- The network is critical infrastructure, not a research project
- It *should* be hard to change infrastructure used to run emergency services and healthcare systems, manage physical utilities and industry, coordinate distribution of goods, and run financial systems

The network can evolve by reinterpreting the protocol stack

- Bits on the wire must be retained, as must aspects of protocol semantics that are visible to middleboxes
- End-to-end behaviour that doesn't impact the network can change – and should change to ensure change is still possible
- TCP is not sacrosanct – neither semantics nor API
- Layer boundaries can change; packet formats can be reinterpreted

Possible directions

- Deprecate UDP as a user-visible protocol, reuse as a *transport identification sub-header* to provide a new extension point ("X tunnelled over UDP" is a kludge; "dynamically signalled X" is an opportunity)
- Relax TCP semantics and API
  - When data is presented to the application is not constrained by the network
  - Segments headers with missing sequence numbers must be retransmitted, but their content can change
  - Reliability, framing, ordering, speed of transmission, and flow control can be modified

UDP Encap is Hard (to specify)

- Jan 2014: MPLS-in-UDP draft IETF Last Call – BIG fail:
  - Concern: IPv6 zero UDP checksum usage
    - Reminder: No IPv6 header checksum, covered by UDP checksum
  - Concern: Congestion (UDP goes everywhere, MPLS doesn't)
- Now (+1 year): Concerns addressed by design team, **BUT**
  - Too much work, took too long, results not directly reusable
  - Approach won't scale up to widespread use of UDP encap
- Ideas for doing better:
  - Additional UDP encapsulation usage and application guidance
    - UDP encap for dummies draft? (e.g., new checksum guidance)
  - Reusable designs (e.g., ECN-based circuit breaker)

User-level Network Stacks

- Argument: place network stacks in user-level in order to ease deployment of new protocols
  - But can it be done while keeping the security and performance provided by in-kernel stacks?
- Our system: MultiStack

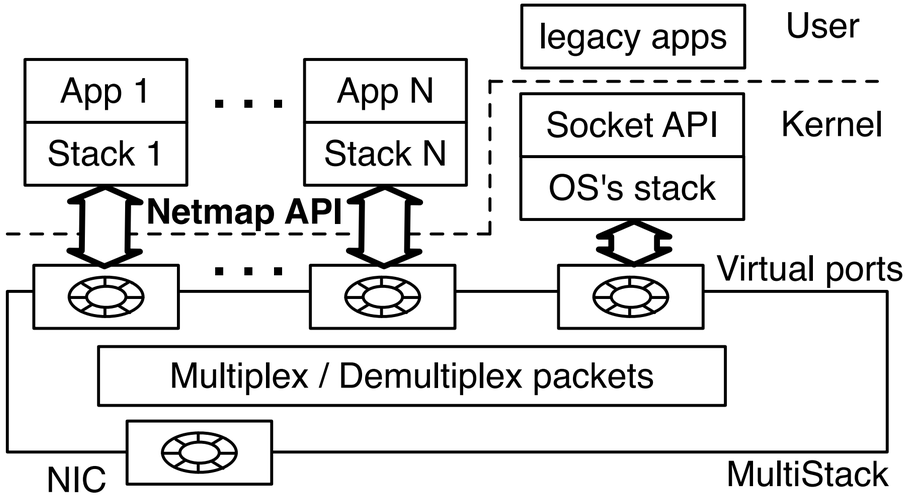


Figure: MultiStack architecture

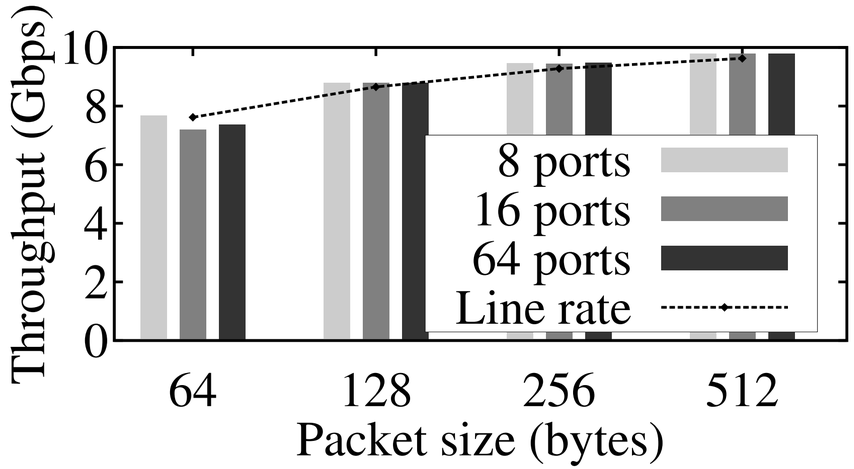


Figure: MultiStack Rx performance

The Much-Maligned TCP

- Widely repeated misinformation about TCP's imagined deficiencies encourages the belief that application developers need to design their own transport protocols layered on top of UDP
- Besides, designing your own transport protocol is fun!
- If everyone else is designing their own transport protocol layered on top of UDP, there must be a good reason for it, right? We should do the same.

Common Application Design Process

1. Build initial prototype using TCP
2. Experience performance (or similar) problem
3. Fail to understand problem; blame TCP
4. Spend six months designing ad hoc protocol
5. Experience same (or worse) problems
6. Admit we just wasted six months? Are you kidding? Ship it anyway; blame performance problems on "bad customer network" or "Wi-Fi packet loss".

Let's not foster ossification

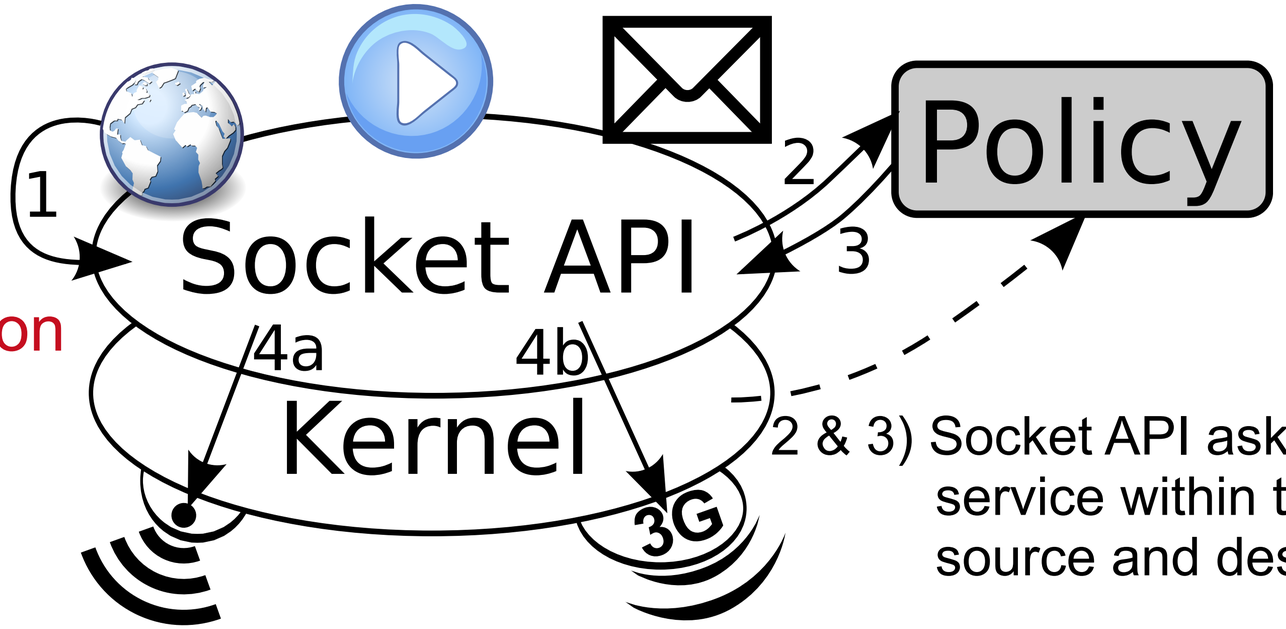
- Transport Services (TAPS) WG decouples applications from transport protocols
  - apps only specify service; transport layer can use/try various protocols
- These days, much focus on minimum-element-that-works-absolutely-everywhere
  - We argue: great to have, but only as a fall-back!
  - Not every path has a middlebox that interferes with the transport layer
- Middleboxes will continue to do their job (see: recent "oh, we can't detect MPTCP, that's a security problem!" concerns)
  - If we make TCP-port-80-with-correct-HTTP-header-and-no-porn-in-the-URL lookalikes the primary choice, middleboxes will have to look even deeper into these packets... not a good direction.

Socket Intents

Let applications tell what they know

1) Application opens a connection providing destination and what it knows about the new connection, e.g.

Flow Category
File Size
Timeliness
Resilience
Bitrate / Duration



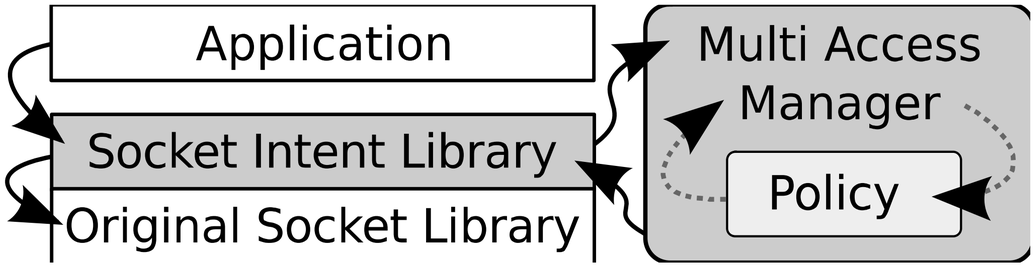
2 & 3) Socket API asks a policy service within the OS to select source and destination address

4a/b) Connection is established using the most appropriate interface (best effort) and gets parameterized respectively if possible

Socket Intents Prototype

Design of our prototype implementation

- Implemented as Socket Library proxy, modifying
  - socket()
  - getaddrinfo()
  - get/setsockopt()
  - bind()
  - connect()
  - close()
  - a new call combining getaddrinfo / socket / setsockopt / connect



- Socket Intents are realized as new sockopt level
- Select destination address from multiple DNS queries
- Select source address by binding the socket

Cross-Layer Coordination

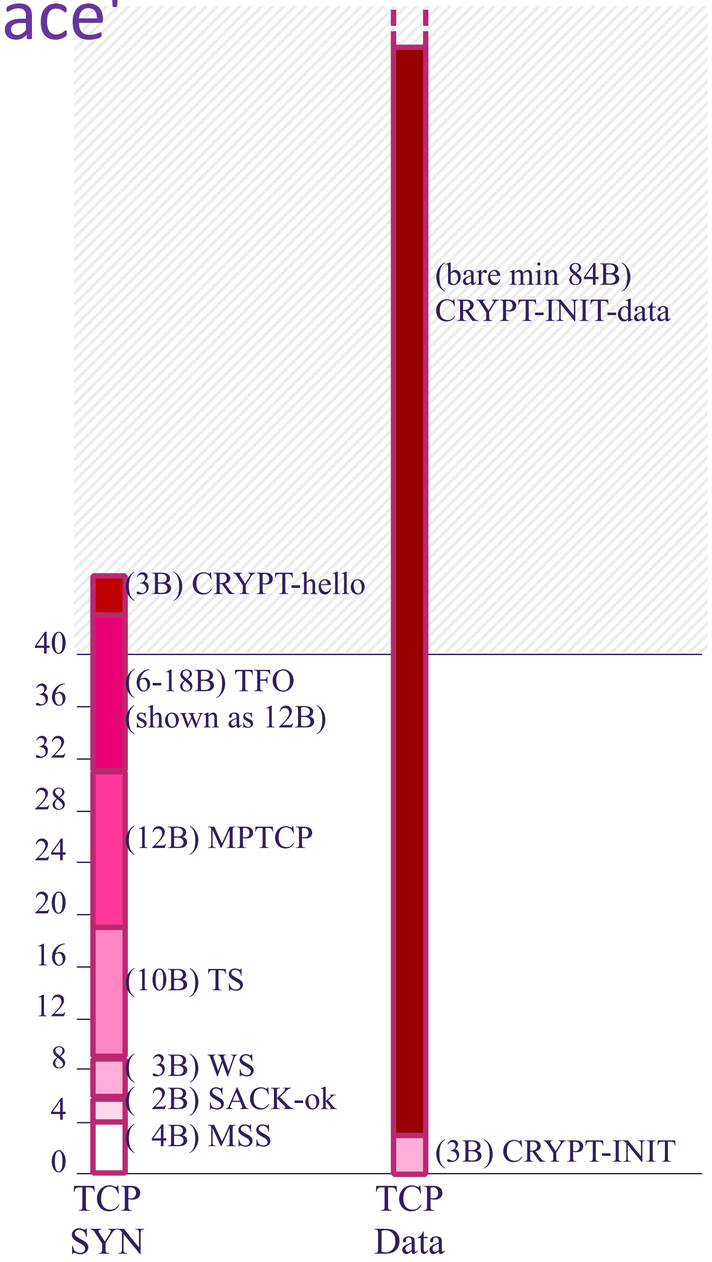
Outlook: Other intentions we can share

- Indicate access network load to
  - place flows on suitable access network
  - shift non time-critical data transfers
  - allow applications to adopt reasonably

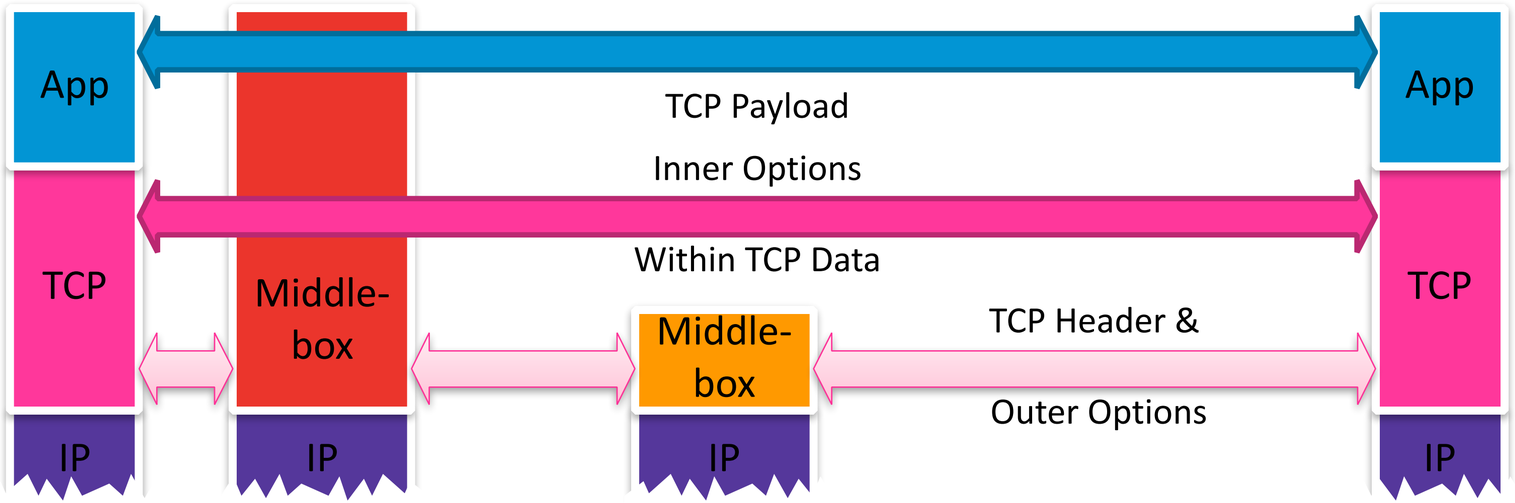
- Advertise service proxies and let the OS choose whether to use them (instead of middle boxes)

- Make CDN and Provider intentions a base for application behavior.

TCP Option 'Space'



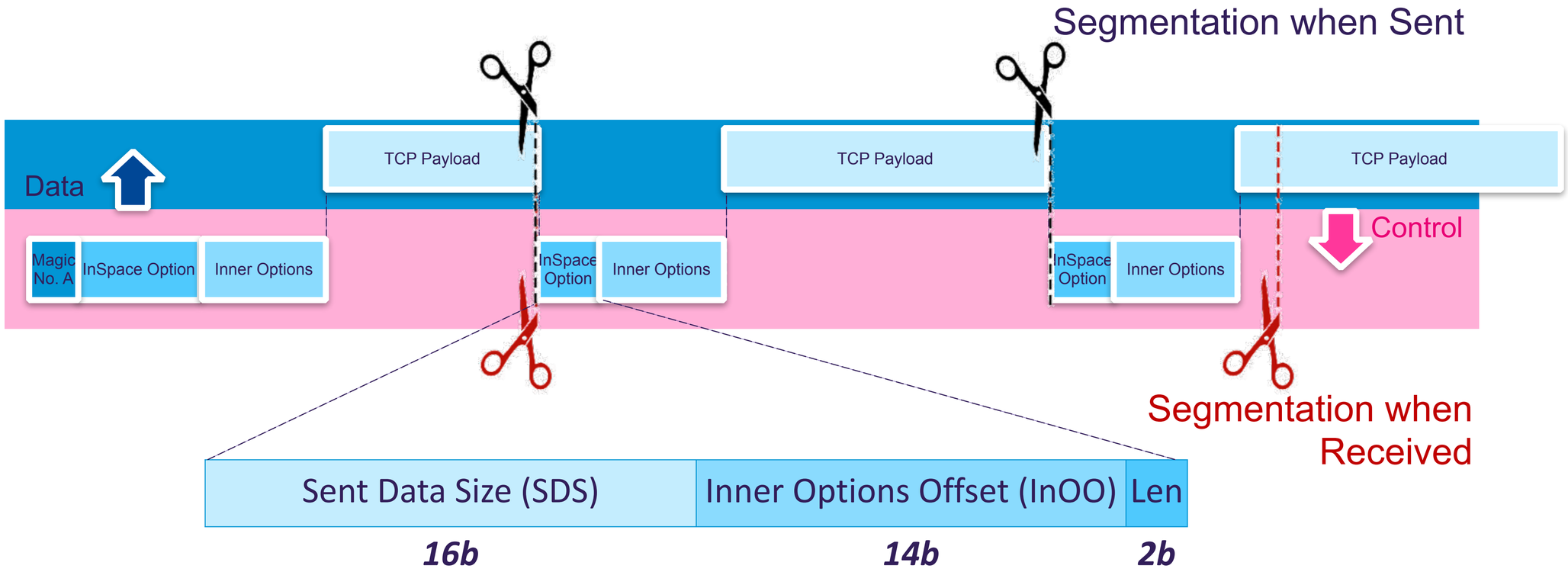
Approach: Tunnel through Inner Space

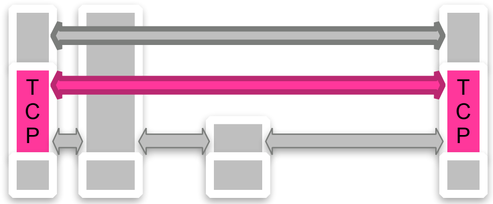


Strawman principle: extend the layer *X* header within layer *X+1* *

* In Internet arithmetic, 4+1 = 7

Inner Space: in the TCP datastream





- robust to resegmentation
- Inner Options not prone to stripping
- in-order delivery of Inner Options
- out-of-order delivery also available

middlebox domination strategy

- if TCP options were MAC'd today, would fail on 10-40% of paths
  - the ends would break a working service

- Inner Space + option authentication (breaks 0%)
- then, if middleboxes move into the TCP data
  - the middleboxes break a working service

why shoot yourself in the foot
when you can make them shoot themselves in the foot?

Backup

tcptrace is your friend

- The tcptrace tool is invaluable for understanding network performance issues
- Too many developers are simply unaware of it
- Custom ad hoc protocols can't benefit from tools like tcptrace, making it even harder to diagnose the problems they encounter

Conclusions

- Need better awareness of how TCP actually works
- Need better awareness of tools like tcptrace to help application developers diagnose and fix problems

Inner Space: Implications & Status

draft-briscoe-tcpm-inner-space-sink-00c
(splitting into sub-drafts - in progress)

draft-briscoe-tcpm-inner-space-01

Payload	Control Options		
	in-order	*out-of-order*	*both*
in-order	*Default*	*(TCP)*	*TCPbis*
out-of-order		*(UDP)*	*UDPbis*
both		*(SCTP)*	*'TCP2'*

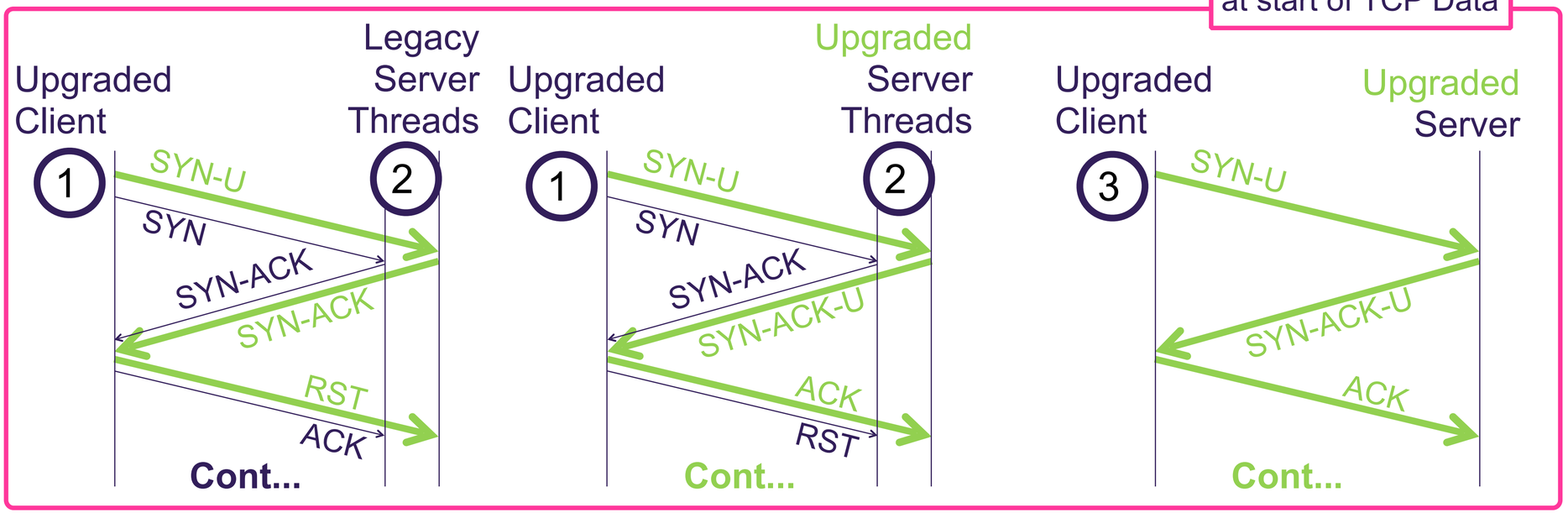
Assessing whether 'TCP2' could satisfy HTTP2 reqs

- Switchable transport semantics
  - Looks like vanilla TCP on the wire
  - switch inner semantics with TCP options e.g. ordering, encryption, compression
  - think "extensible Minion"
- Example: tcpcrypt decomposition
  - cut from 18 to 9 CRYPT sub-options
  - removed handshake latency
  - can encrypt control options, and MAC pure ACKs
- Progress since Jul'14
  - Default mode: Full spec as individual draft (5 revs, presented in tcpm & tcpinc)
  - TCPbis mode: Full spec available but not submitted <http://bobbriscoe.net/projects/2020comms/tcp/draft-briscoe-tcpm-inner-space-sink-00c.txt>
  - ad hoc team formed (~20 people on mailing list)
  - half-a-dozen doing or planning path traversal testing
  - 2 or 3 planning to implement, including upstreaming

dual handshake... and migration to single

1. different source ports, same dest. port
2. no co-ordination needed between server threads
   can be physically separate replicas

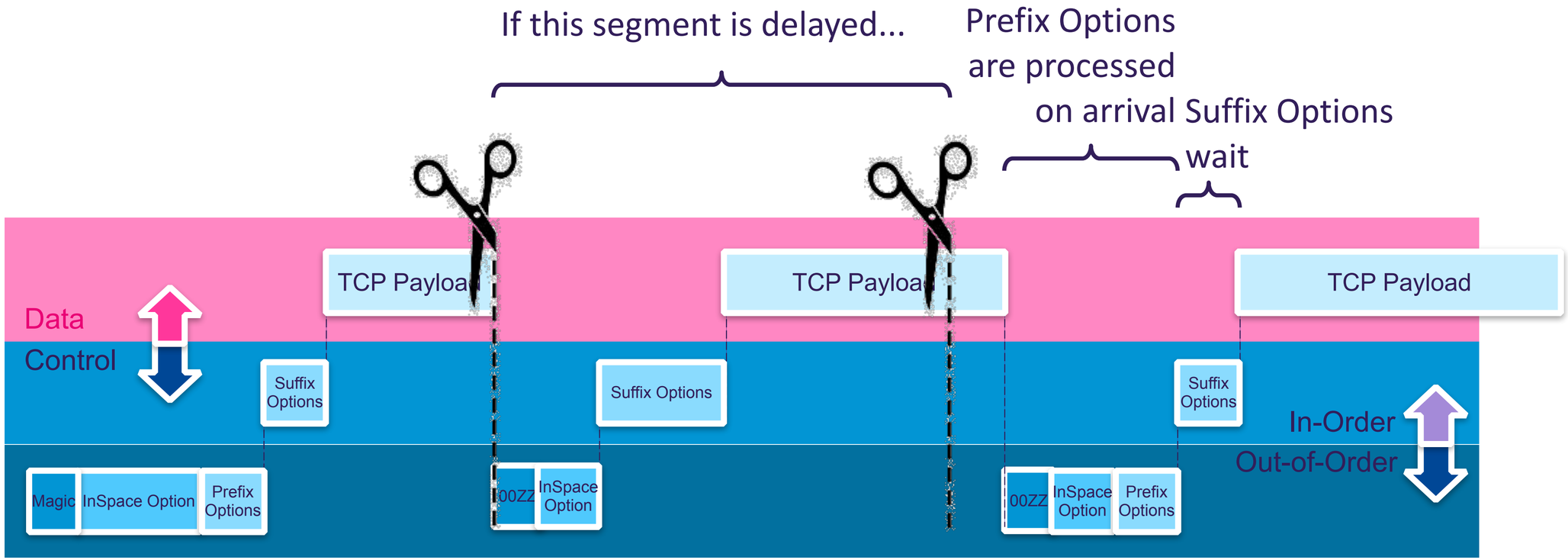
-U = upgraded,
i.e. magic no.
at start of TCP Data



3. Can use single SYN-U handshake
   - when server is in cached white-list
   - once deployment is widespread (no need for white-list)

   Fall-back to SYN if no SYN-ACK-U

TCPbis mode: 2 control channels in the datastream



- Rcvr can reconstruct sent segments - robust to resegmentation
- TCP has always processed Outer Options on arrival (out-of-order)
- Inner Space adds two types of Inner Option to avoid middlebox interference
  - In-order Suffix Options – for stream control
  - Out-of-order Prefix Options
    - essential for a few ACK-related options* to avoid flow-control deadlock

* SACK, MPTCP Data ACK, tcpcrypt MAC of ACK