Inner Space

for tcpinc

Bob Briscoe

Nov 2014

draft-briscoe-tcpm-inner-space-01

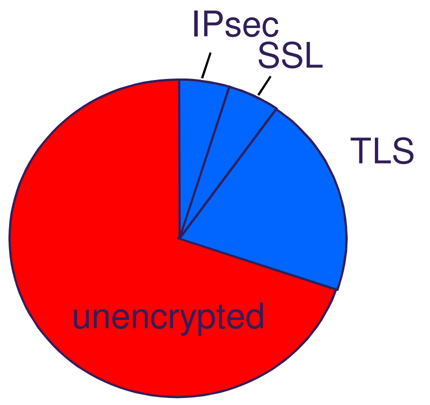
Bob Briscoe's work is part-funded by the European Community
under its Seventh Framework Programme through the
Trilogy 2 (ICT-317756) and the RITE (ICT-317700) projects

opportunistic encryption

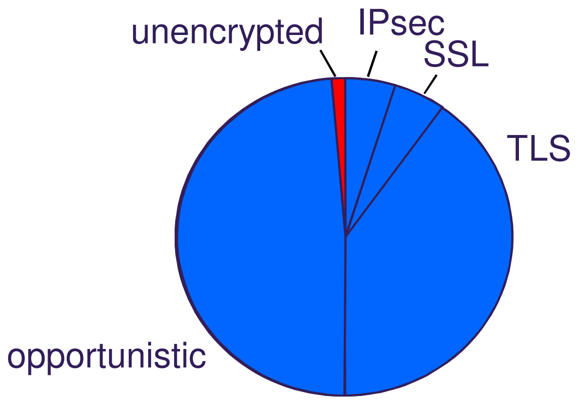
- tcpinc depends on TCP options
- TCP options are changing...
  - special session in tcpm on 4 drafts

1. No handshake latency
2. Middlebox not a downgrade
3. How? Inner Space protocol
4. Authentication coverage insights

Reminder: Project Goal

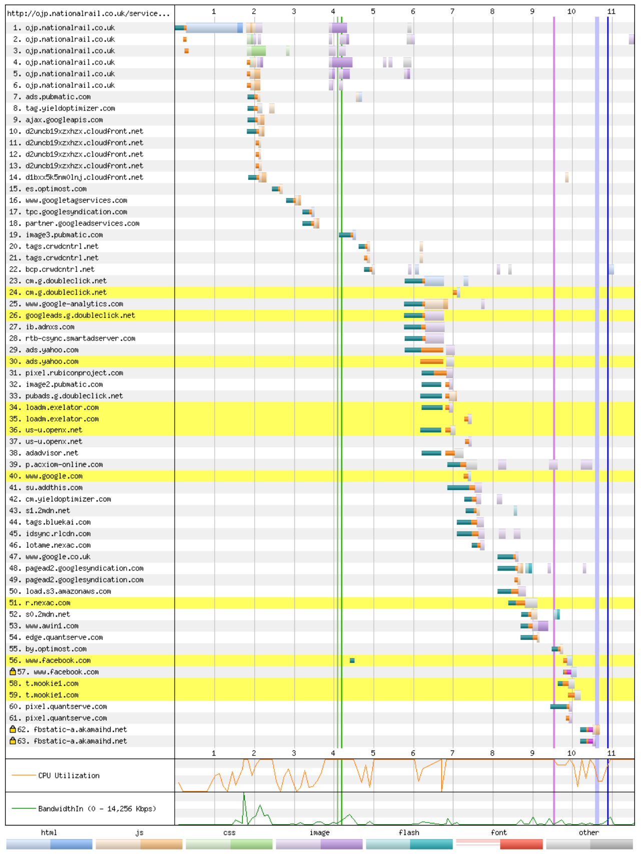


TCP traffic today



Goal for TCP traffic

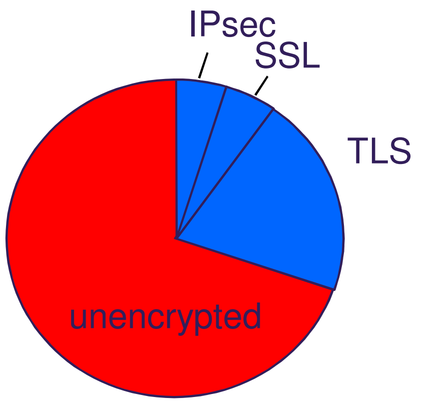
opportunistic delay



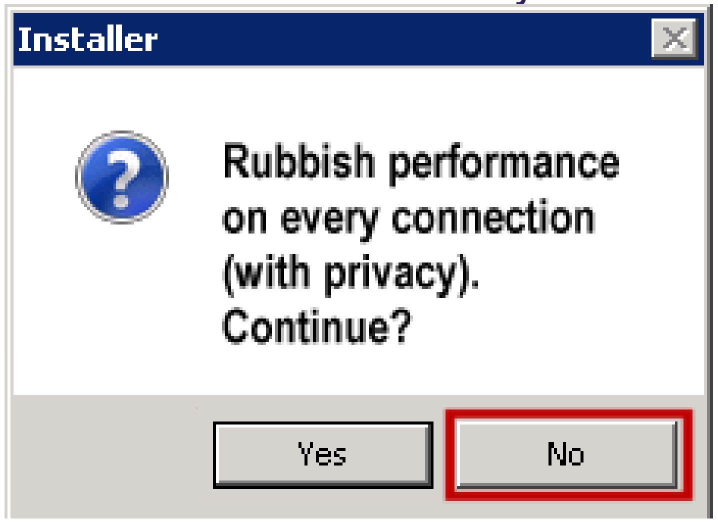
Ave. Alexa 1000 Web page
44 TCP connections*
from 15 servers†

* as of Nov 2014
† as of early 2013
httparchive.org

Reminder: Project Goal

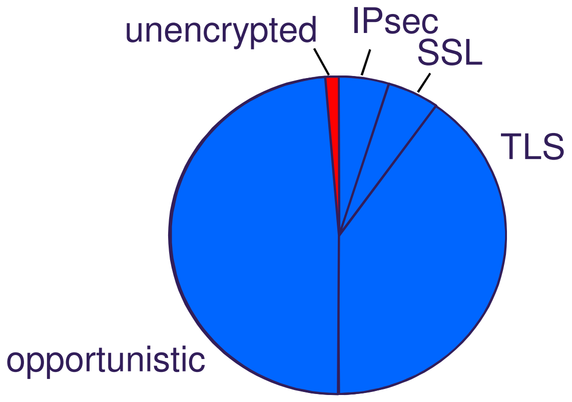


TCP traffic today

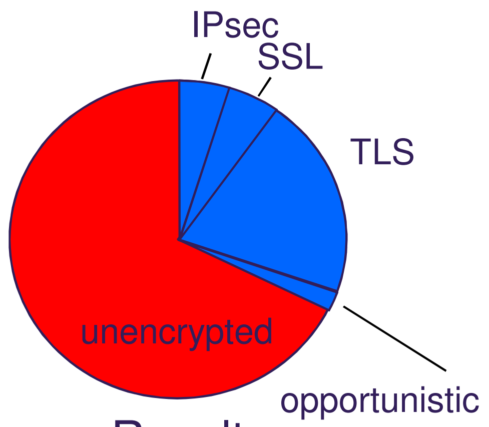

Installer

Rubbish performance on every connection (with privacy). Continue?

Yes [✓ No]



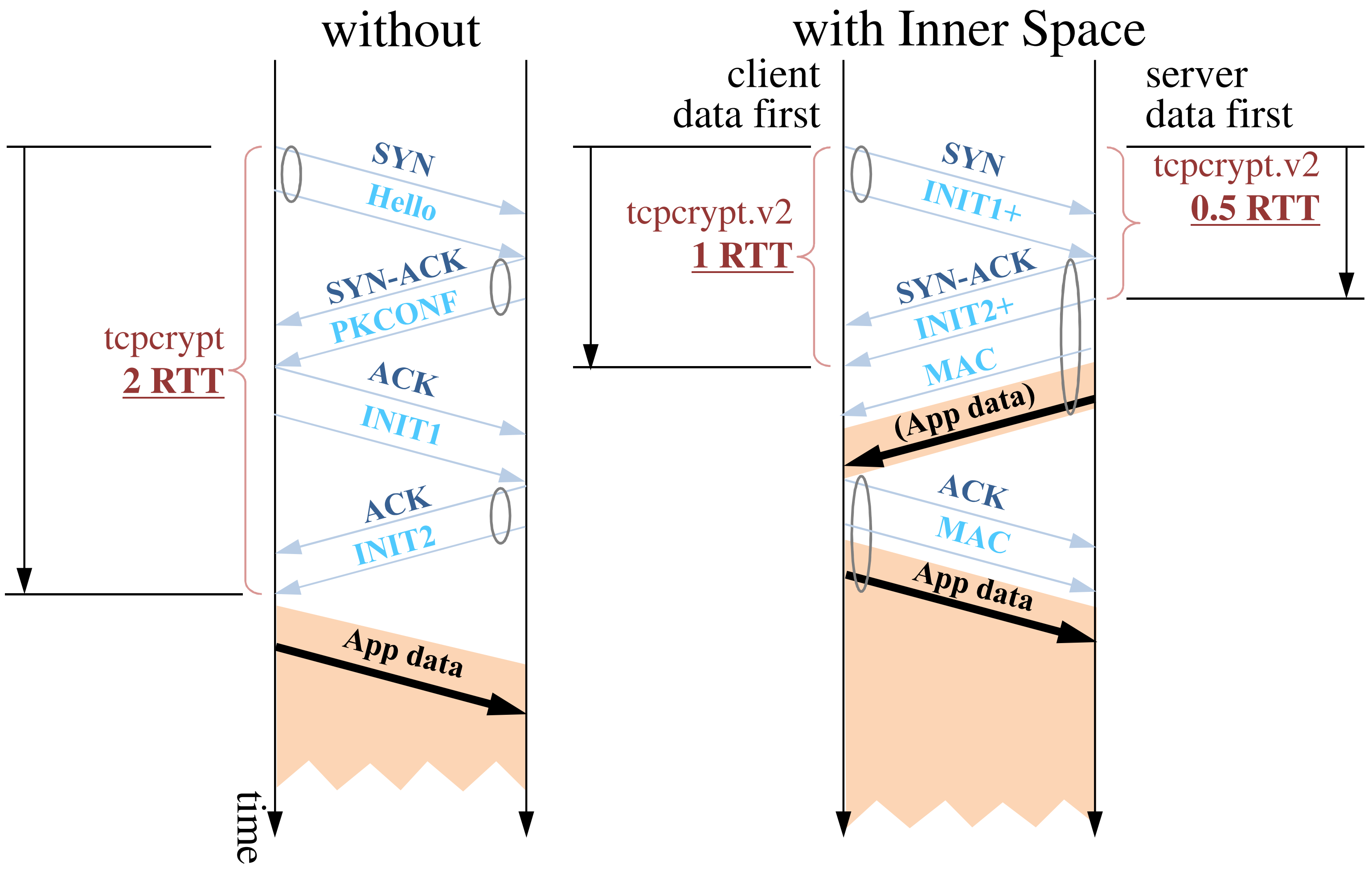
Goal for TCP traffic



Result

tcpcrypt latency & Inner Space

session initialisation



Legend:
loop **groups messages that can be sent in the same packet**

TCP header
TCP options
App data

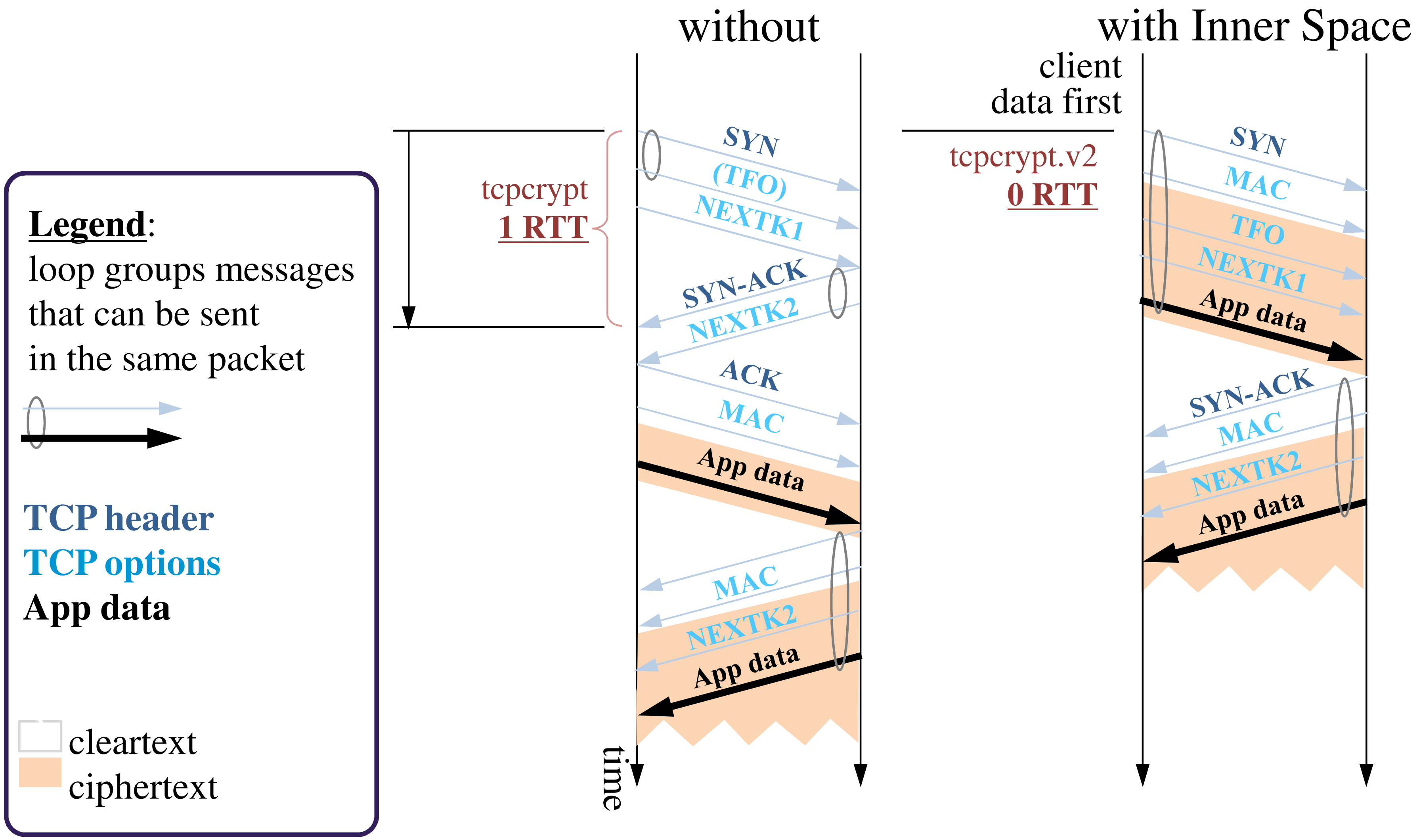
⬜ cleartext
🟧 ciphertext

INIT1, sym-cipher-list, N_C, pub-cipher, PK_C	**INIT1+, pub-cipher-list, sym-cipher-list, N_C, PK_C**
INIT2, sym-cipher, KX_S	**INIT2+, sym-cipher, KX_S; MAC<m>**

- cuts out two states
- decouples tcpcrypt from TCP state machine?

tcpcrypt latency with Inner Space

session resume



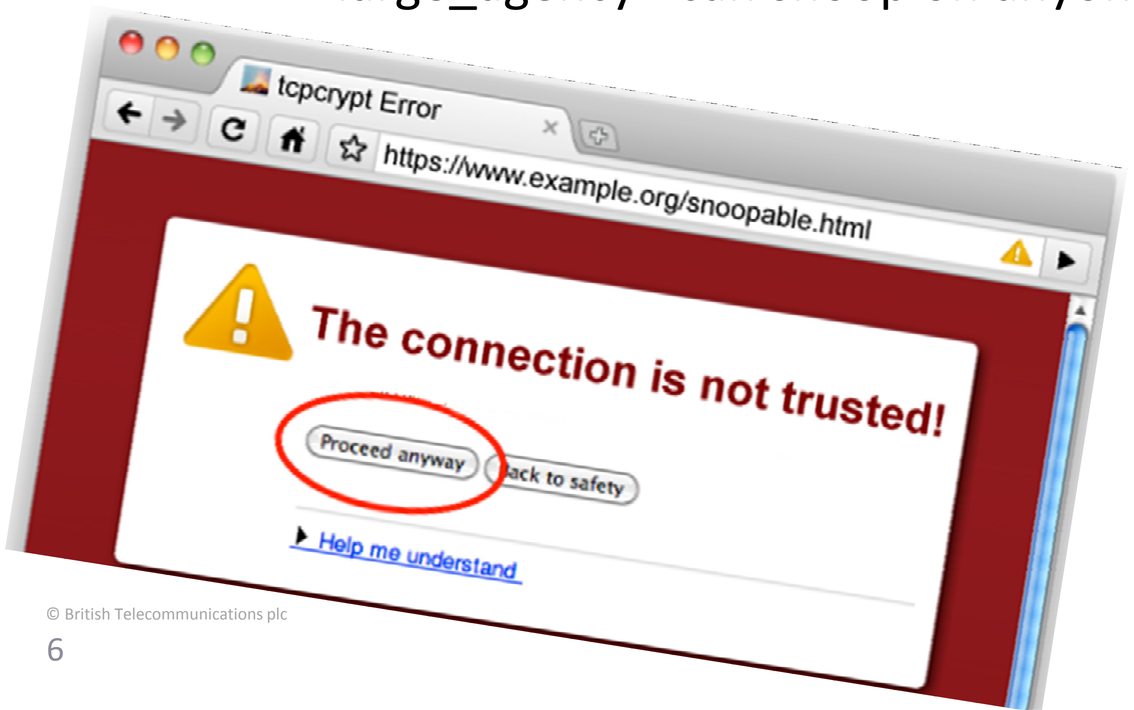
- see [Briscoe14] for details

middleboxes: detect-and-downgrade?
not good enough

unknown TCP option stripped	
to port	% paths
80 (HTTP)	14%
443 (HTTPS)	6%
34343 (unassigned)	4%

[Honda11]

- tcpinc (tcpcrypt and TCP-TLS) relies on new TCP options
  - so tcpinc would disable itself on ~10% of paths
  - when middlebox downgrade of tcpinc is so *unremarkable*
    it makes a downgrade attack indistinguishable from a middlebox
    - <large_agency> can snoop on anyone


tcpcrypt Error
https://www.example.org/snoopable.html
The connection is not trusted!
Proceed anyway
Back to safety
Help me understand

middlebox domination strategy

long term aim

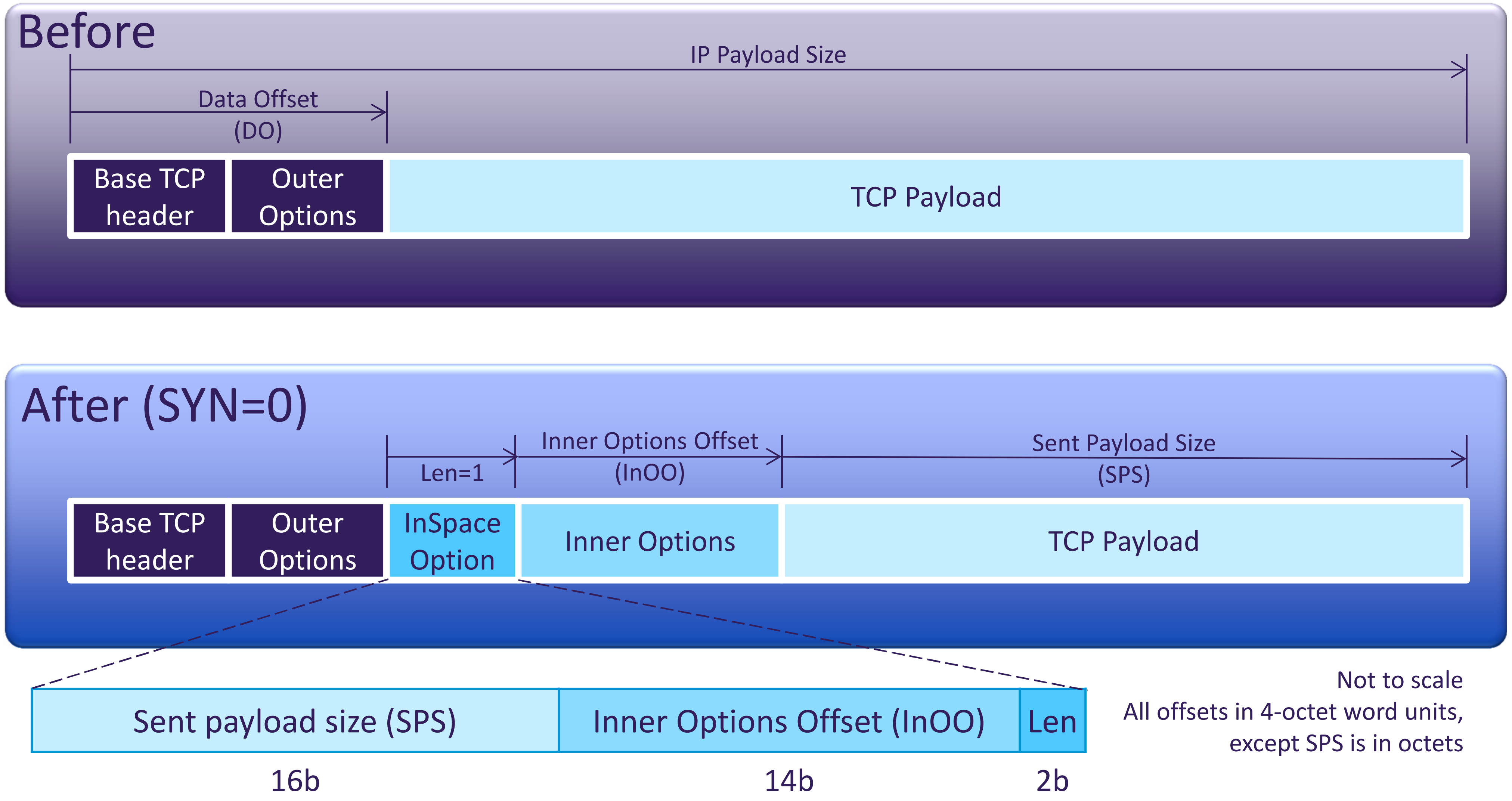
- authenticate options

- if turned on option authentication today
  - ~10% of connections would break
  - the ends break a working service

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

if you want to shoot them,
why shoot yourself in the foot
when you can make them shoot themselves in the foot?

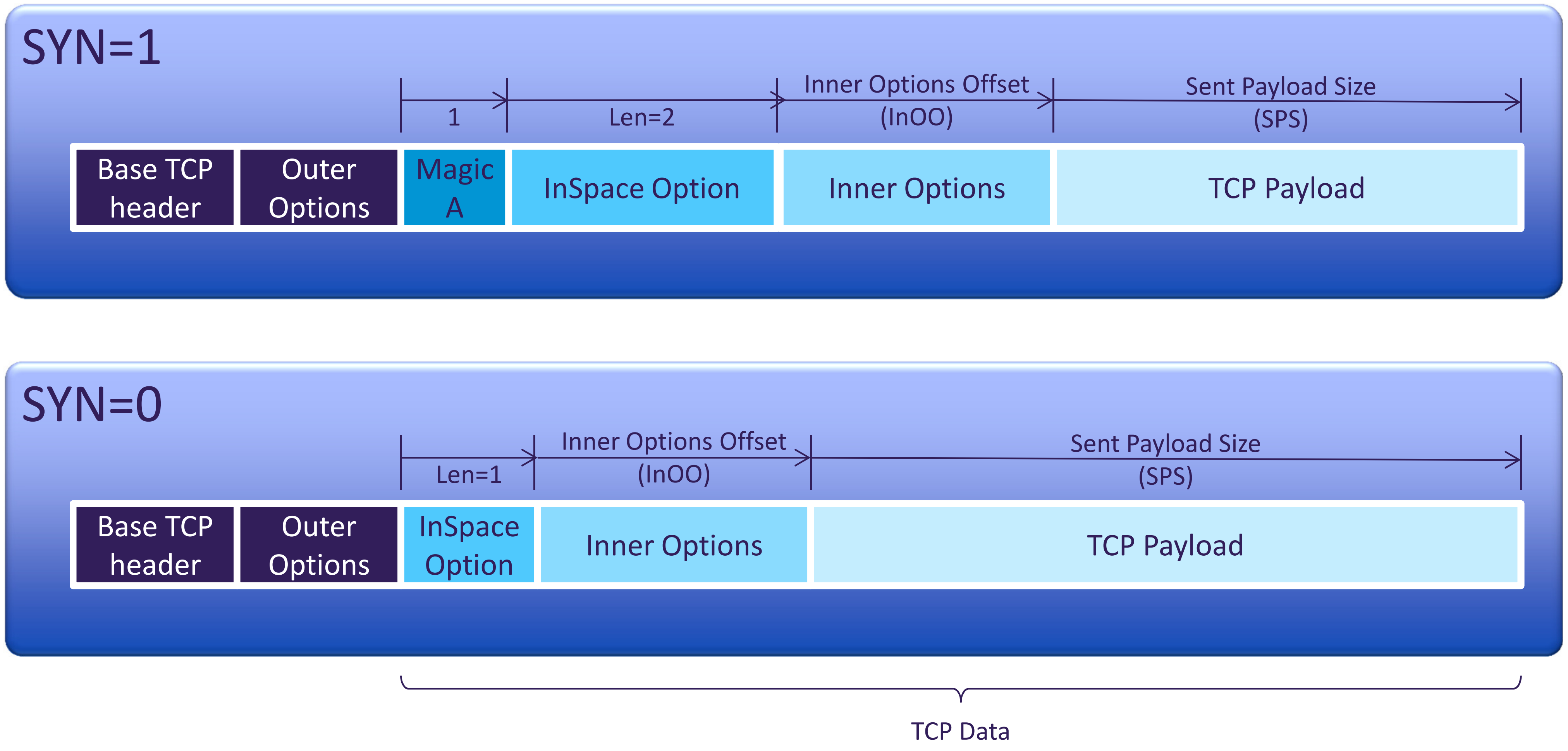
Inner Space – TCP segment structure (SYN=0)



Not to scale
All offsets in 4-octet word units,
except SPS is in octets

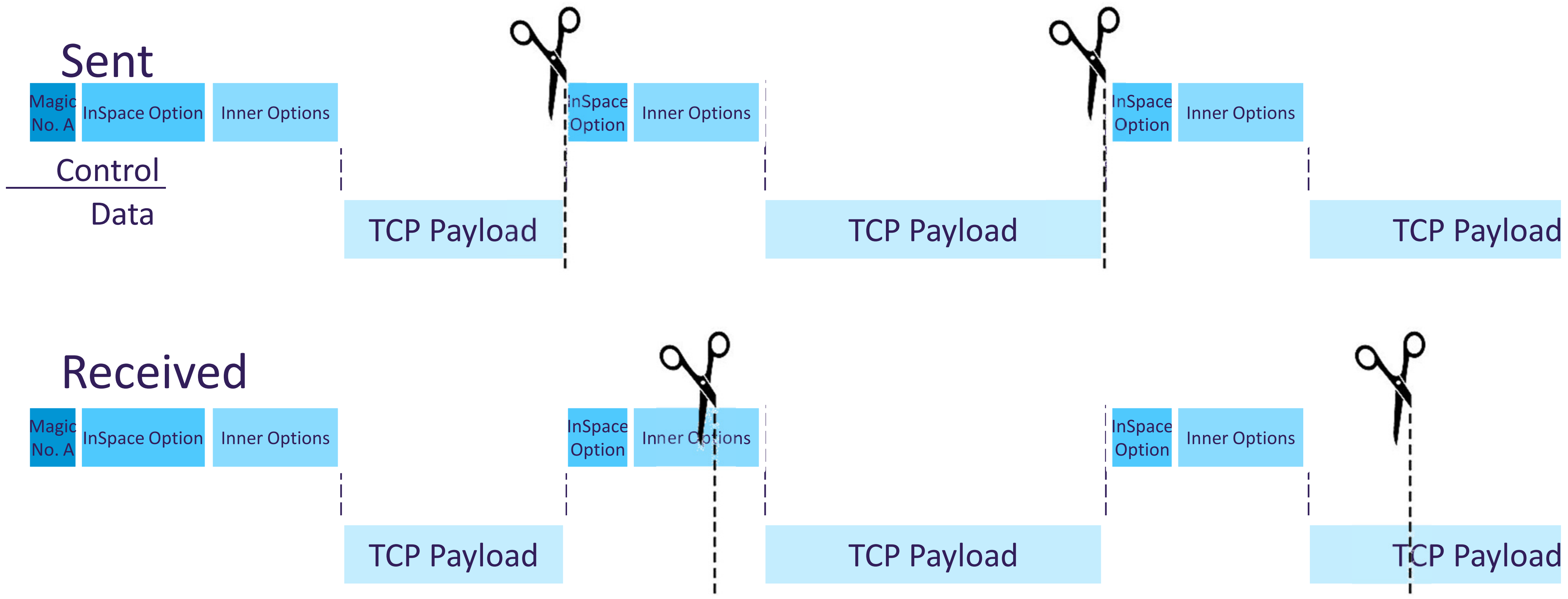
- InSpace is solely a framing header

Inner Space – TCP segment structure



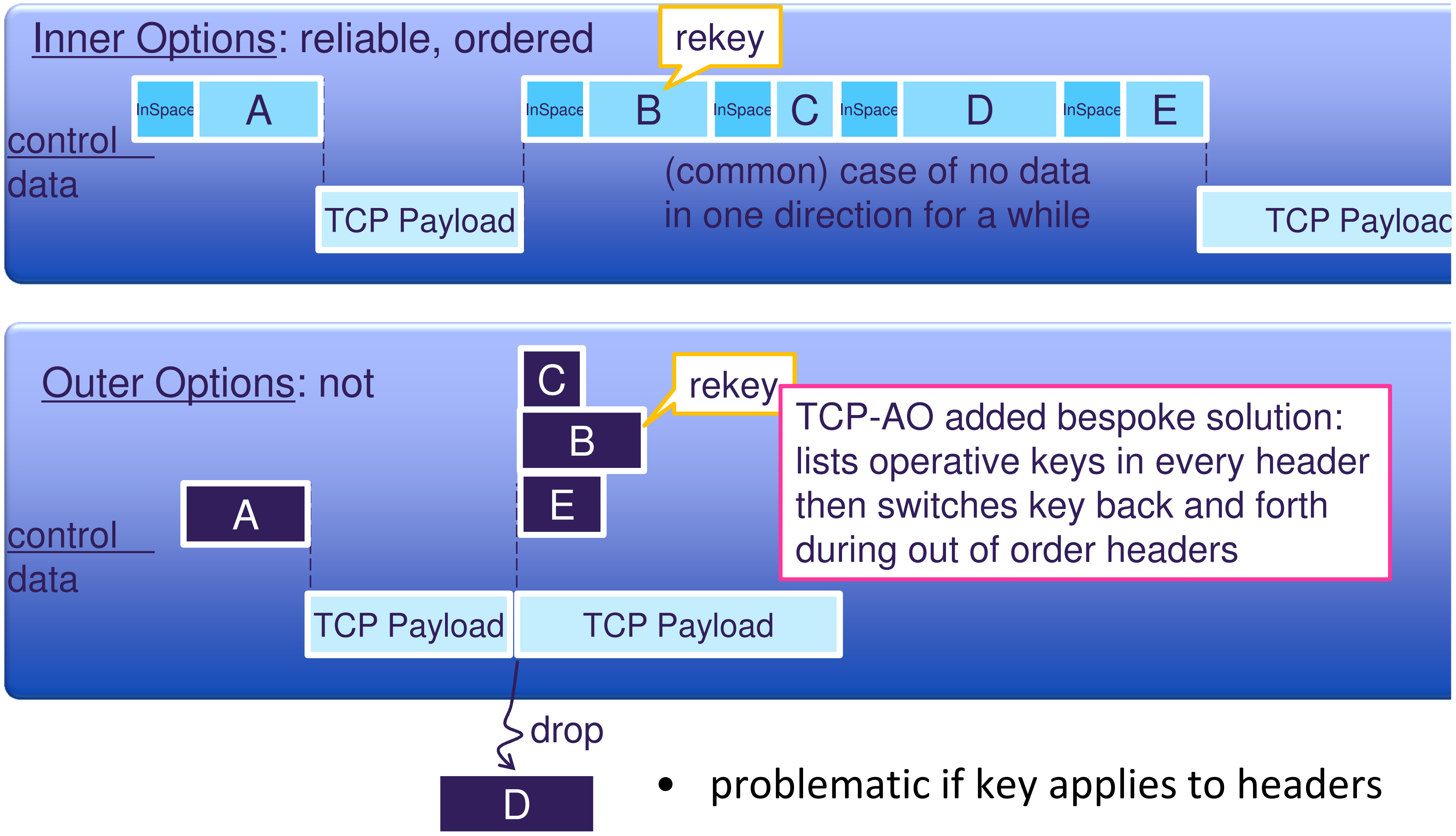
- presence of Inspace flagged by magic no. at start of each stream
- avoided an Outer TCP Option as the flag, which could be stripped
- inherently safe to flag within the payload – shares fate with options

Inner Space – TCP byte-stream



- robust to resegmentation
- Inner Options not prone to stripping
- reliable ordered delivery of Inner Options
  1. makes rekey easy (gives tcpcrypt TLS-like records)
  2. tcpinc can encrypt Inner Options (incl. its own)

rekey message on an unreliable unordered segment



TCP-AO added bespoke solution: lists operative keys in every header then switches key back and forth during out of order headers

- problematic if key applies to headers

transformation of the datastream

controlled by TCP options within the datastream

- e.g. (de)crypt, (de)compress
- care with processing order: recursion limited to one level
  - SYN=1:
    - if not previously found MagicA, retry after transformation

  - SYN=0:
    - (de)crypt progressively
    - up to the end of each set of Inner Options
    - process those options
    - then continue with next segment (might be with a new key)

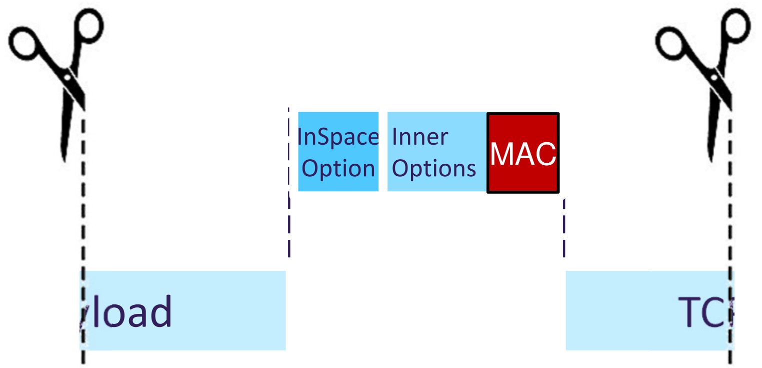
transformation of the datastream

controlled by TCP options within the datastream

- e.g. (de)crypt, (de)compress
- care with processing order: recursion limited to one level
  - SYN=1:
    - if not previously found MagicA, retry after transformation

  - SYN=0:
    - (de)crypt progressively
    - up to the end of each set of Inner Options
    - process those options
    - then continue with next segment (might be with a new key)

message authentication coverage



- coverage options [Marcelo BB]:
  1. payload only
  2. payload plus some header fields
     a) MAC in a TCP option
     b) MAC in the payload
        – possible exception: MAC for pure ACKs in TCP option
     c) MAC for header in a TCP option; and for payload in payload
     + MAC in a TCP option... in the payload

- Inner Space preserves the 1-1 mapping between
  – MAC, payload & Inner TCP options of each segment
  – but not Outer Options and not the main TCP header (next slide)

- gotcha: MAC consumes sequence space on pure ACK
  – could write ad hoc rules, e.g. "defer ACK if no payload"
  – full solution (next revision): unreliable & reliable Inner Options

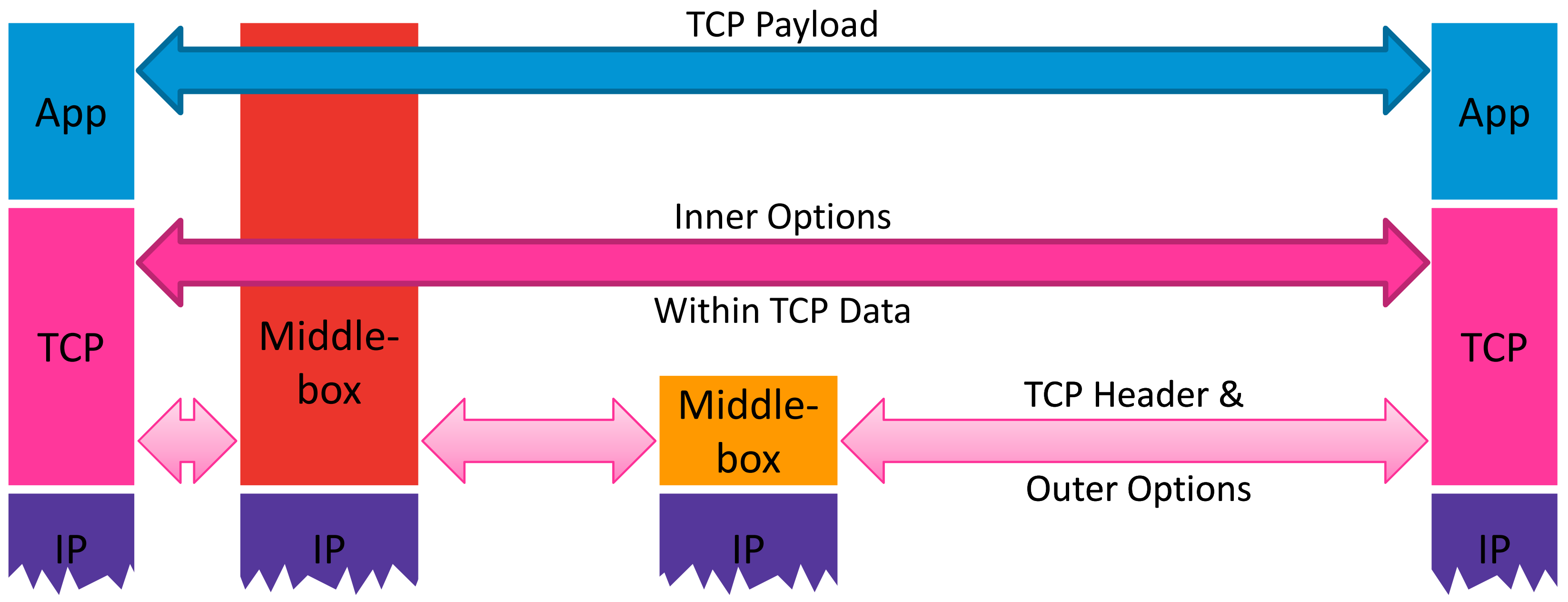
message authentication of the main TCP header

- middlebox transformations
    - NAT
    - resegmentation
    - Seq No Shift
    - ACK thinning
    - <more will be discovered>
- not really attacks, but naive authentication would fail
    - approach so far: absolve these headers from authentication
- a feasible approach (not universally applicable)
    - using inner options, sender reveals the original* once per connection
    - rcvr reverses shifts, reconstructs sent (pseudo)segments
    - rcvr verifies sent MAC against reconstructed pseudosegments

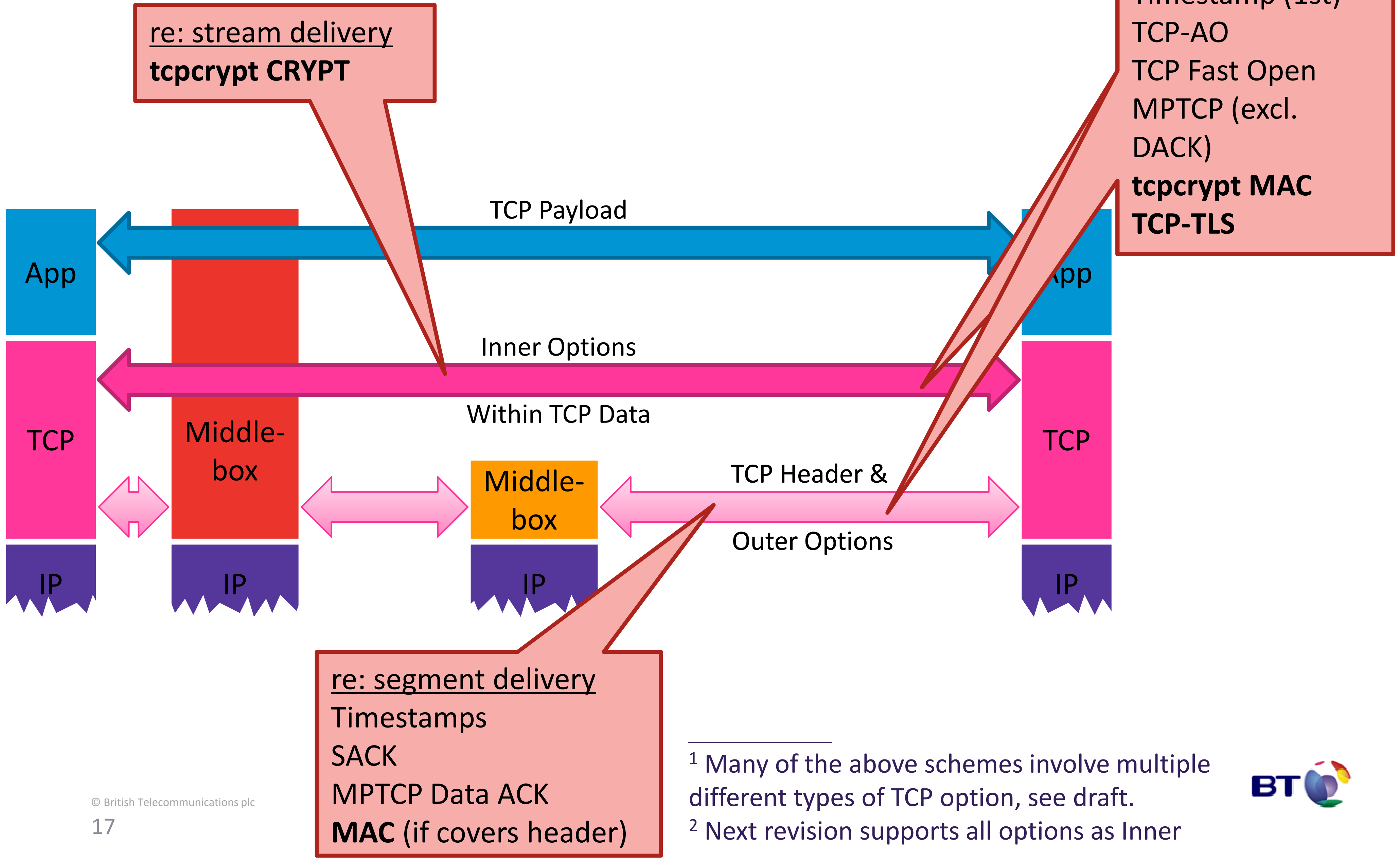
- summary: verify that header transformations are *consistent*

* privacy concern?

Inner Space – encapsulation model



Inner Space – applicability & compatibility[1,2]



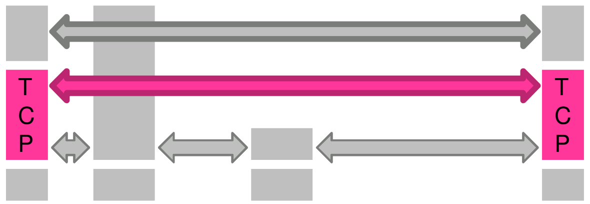
<u>re: stream delivery</u>
tcpcrypt CRYPT

<u>re: connection</u>
Max Segt Size
SACK-ok
Wnd Scale
Timestamp (1st)
TCP-AO
TCP Fast Open
MPTCP (excl. DACK)
tcpcrypt MAC
TCP-TLS

<u>re: segment delivery</u>
Timestamps
SACK
MPTCP Data ACK
MAC (if covers header)

[1] Many of the above schemes involve multiple different types of TCP option, see draft.
[2] Next revision supports all options as Inner

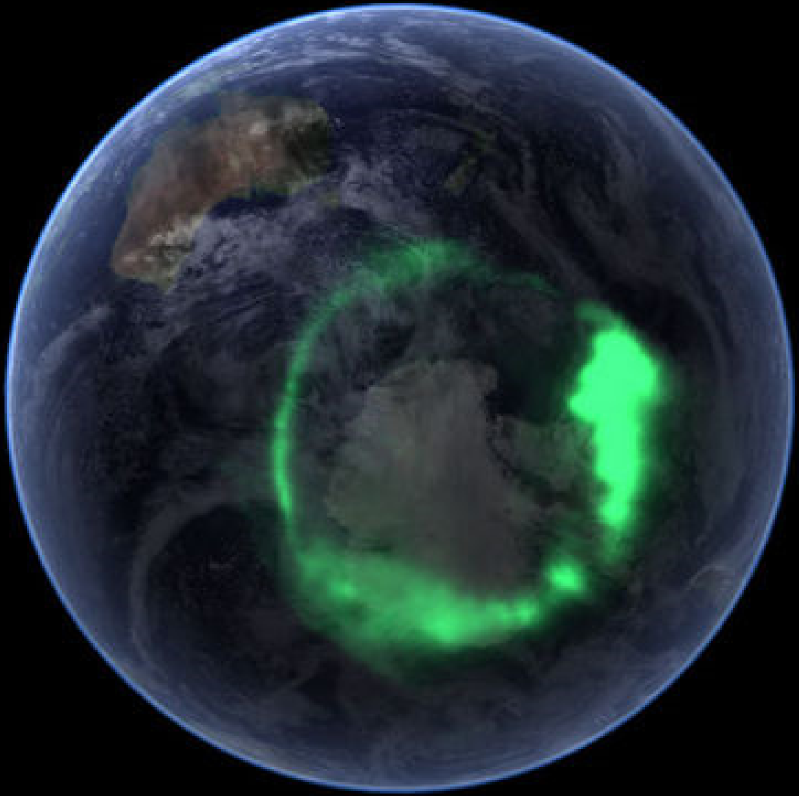
summary



1. much more TCP option space
2. cuts handshake latency ...as a service
3. middlebox traversal ...as a service
4. sent segment reconstruction ...as a service
5. reliable ordered options ...as a service

next steps

- mismatch in maturity?
  - tcpm chairs: "no hurry"
- tcpcrypt.v2 decomposition
  - review pls
- path testing
  - data in SYN, is DPI bypass necessary? viable?
- implementation
  - compatibility testing
- IAB workshop on stack evolution in a middlebox Internet
  - principles



Inner Space

Q&A

Spare slides

contents

1. No handshake latency
2. Middlebox not a downgrade
3. How? Inner Space protocol
4. Authentication coverage insights

Spare slides

1. More info
2. Dual handshake
3. Overhead
4. Extensions – DPI traversal, EchoCookie
5. Tricky bits
6. Interaction with TCP Fast Open
7. Work in progress

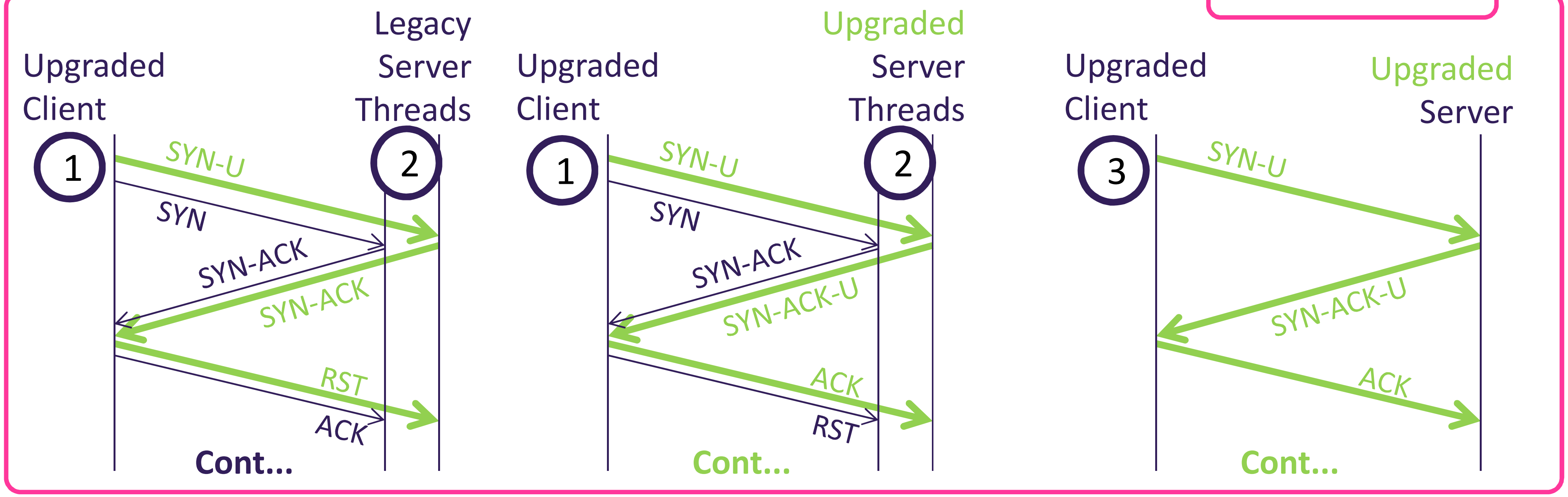
more info

- Inner Space for TCP Options
  - draft-briscoe-tcpm-inner-space
- [Bagnulo14] Protect or not the TCP header fields
  - http://www.ietf.org/mail-archive/web/tcpinc/current/msg00359.html
- [Briscoe14] tcpcrypt decomposition:
  - http://www.ietf.org/mail-archive/web/tcpinc/current/msg00384.html
- [Honda11]
  - 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, November 2011.

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

☹ drawbacks - overheads

			Example
• Dual Handshake			
– Latency	(Upgraded Server)		Zero
	(Legacy Server)		Worst of 2
– Connection Rate	P*D		8%
– Connection State	P*D/R		2.7%
– Network Traffic	2*H*P*D/J	counting in bytes	0.03%
	2*P*D/K	counting in packets	0.2%
– Processing	{pending implementation}		?
• Option on every non-empty segment			
– Network Traffic	P*Q*4/F		0.04%
– Processing	{pending implementation}		?

Example

P : [0-100%] proportion of connections that use extra option space	80%
D : [0-100%] proportion of these that use dual handshake	10%
R : [round trips] ave. hold time of connection state	3
H : 88B for IPv4 or 108B for IPv6 (see draft for assumptions)	
J : ave bytes per connection (in both directions)	50KiB
K : ave packets per connection (in both directions)	70 packets
Q : ave prop'n of InSpace connections that use it after handshake	10%
F : [B] ave frame size	750B

☹ drawbacks - non-deterministic

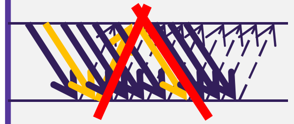
- the magic number approach traverses option stripping middleboxes, but...
- probability that an Upgraded SYN or SYN/ACK is mistaken for an Ordinary Segment: Zero
- probability that an Ordinary SYN or SYN/ACK with zero payload is mistaken for an Upgraded Segment: Zero
- probability that payload data in an Ordinary SYN or SYN/ACK is mistaken for an Upgraded Segment: $<< 2^{-66}$
  (roughly 1 connection collision globally every 40 years)

Extensions – summary of dependencies

- mandatory if implement Inner Space

EchoCookie TCP option

- extensions: optional while Inner Space is Experimental



- ModeSwitch TCP Option (scope wider than Inner Space)

- Explicit Dual Handshake (2 Outer TCP Options)

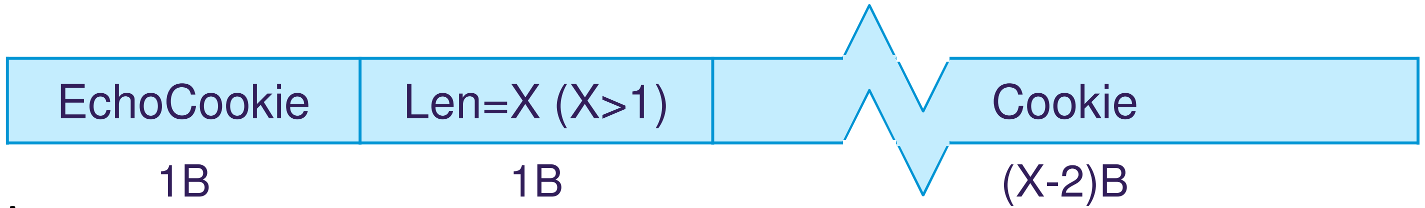
- Jumbo InSpace Option

- Inner Space segment structure for DPI traversal

see spare slides or draft

tricky bits – SYN floods

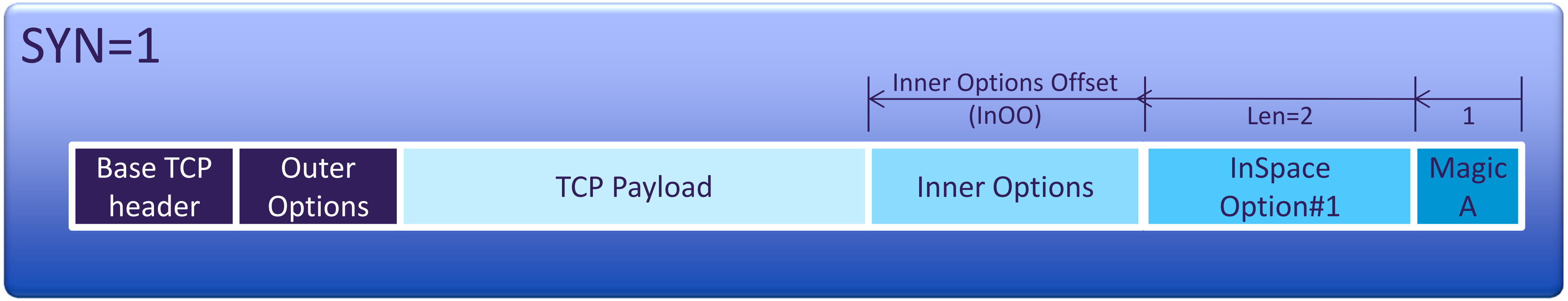
- current SYN cookie mechanism is too small for the ambition to use lots of options
  - because it packs the cookie into part of the Initial Seq No
  - solution: a larger cookie jar that an Inner Space host MUST implement
- the EchoCookie option (can be independent of Inner Space)
  - if host receives a cookie, it MUST reflect it back
  - sender can choose size and contents

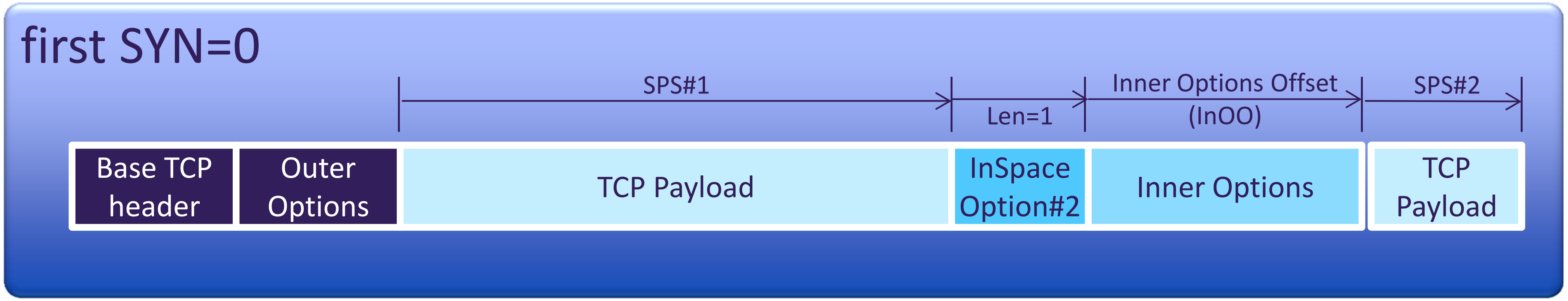


- other opportunities
  - tcpcrypt could use this

extension – DPI traversal

- conjecture: DPI often parses payload & stops when it finds what it needs
- solution?: locate MagicA at the end of the segment
  - server searches for MagicA at end if not at start



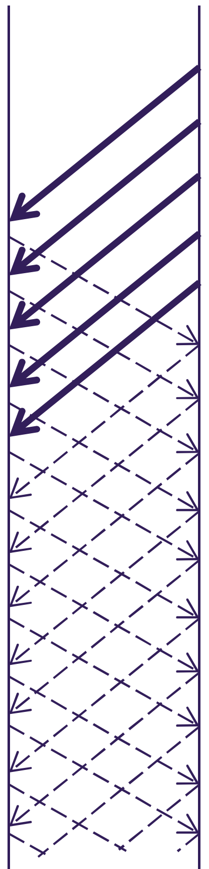


- can't work from the end of every segment, only the first
  - then use the spare first SPS (SPS#1) for the second segment

tricky bits - zero payload segments

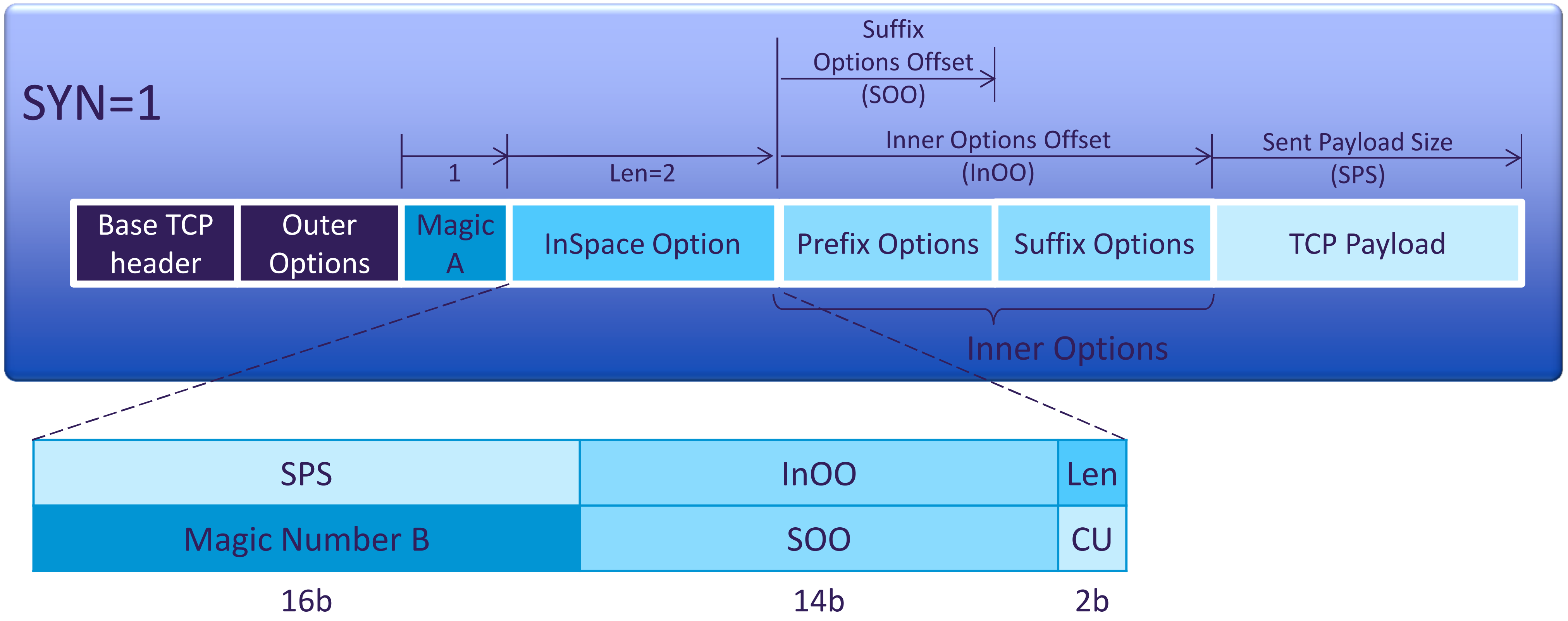
- zero payload segments
  - MAY include an Inner Option
  - SHOULD NOT repeat the same Inner Options until more payload

Without the 'SHOULD NOT' it would continue to ACK ACKs for ever



- other tricky bits → spare slides or draft
  - the EchoCookie for SYN floods
  - retransmissions during handshake
  - explicit dual handshake
    - corner cases of dual handshake
    - deferred data in SYN

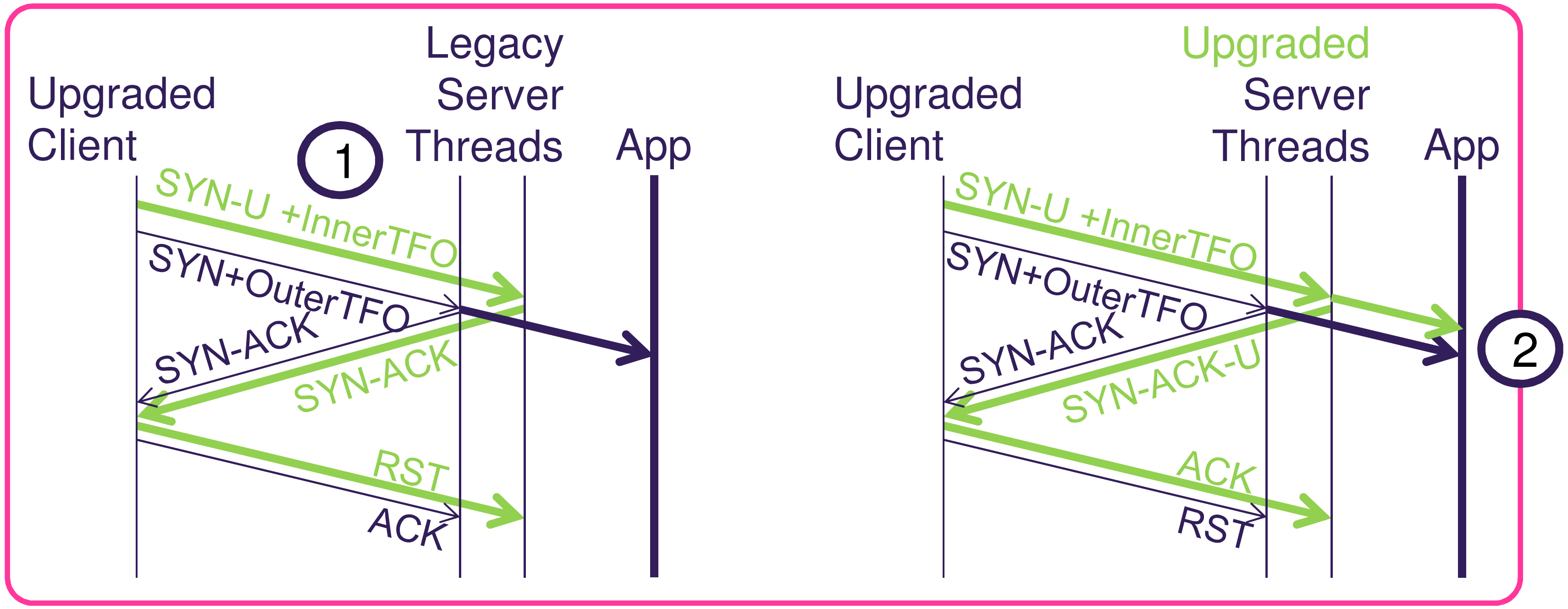
tricky bits – option processing order



- only on the first segment of each half-connection
  - on later segments, Outer Options have to be processed before Inner
  - reason: can't find Inner Options if still waiting to fill a sequence gap

Inner Space & TCP Fast Open (TFO)

1. If Upgraded Client uses TFO
   - MUST place cookie in Inner of SYN-U
   - then Legacy Server will not pass corrupt TCP Data to app before RST

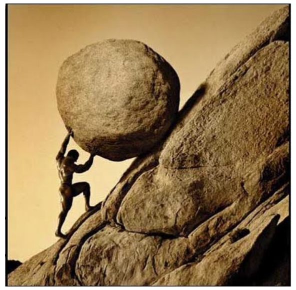


-U = upgraded, i.e. magic no. etc. at start of TCP Data

2. If dual h/s, Upgraded Server will pass payload to app twice
   - OK, because TFO only applicable if app immune to duplication

work in progress

- Unreliable as well as reliable ordered Inner Options



  - without consuming rwnd
  - without consuming sequence space (avoiding middlebox 'correction')
  - delivered immediately in received order, not sent order
  - based on ideas in TCP Minion

  - spec fully written-up – internal review prior to posting