Inner Space

Bob Briscoe

Oct 2014

draft-briscoe-tcpm-inner-space-01

menu

- Status & purpose of presentation
- Inner Space protocol
- Benefits & drawbacks
- Tricky bits
- Extensions (in spare slides)
- Applicability / compatibility
- Opportunities / further work
- Next steps

BT

purpose & status

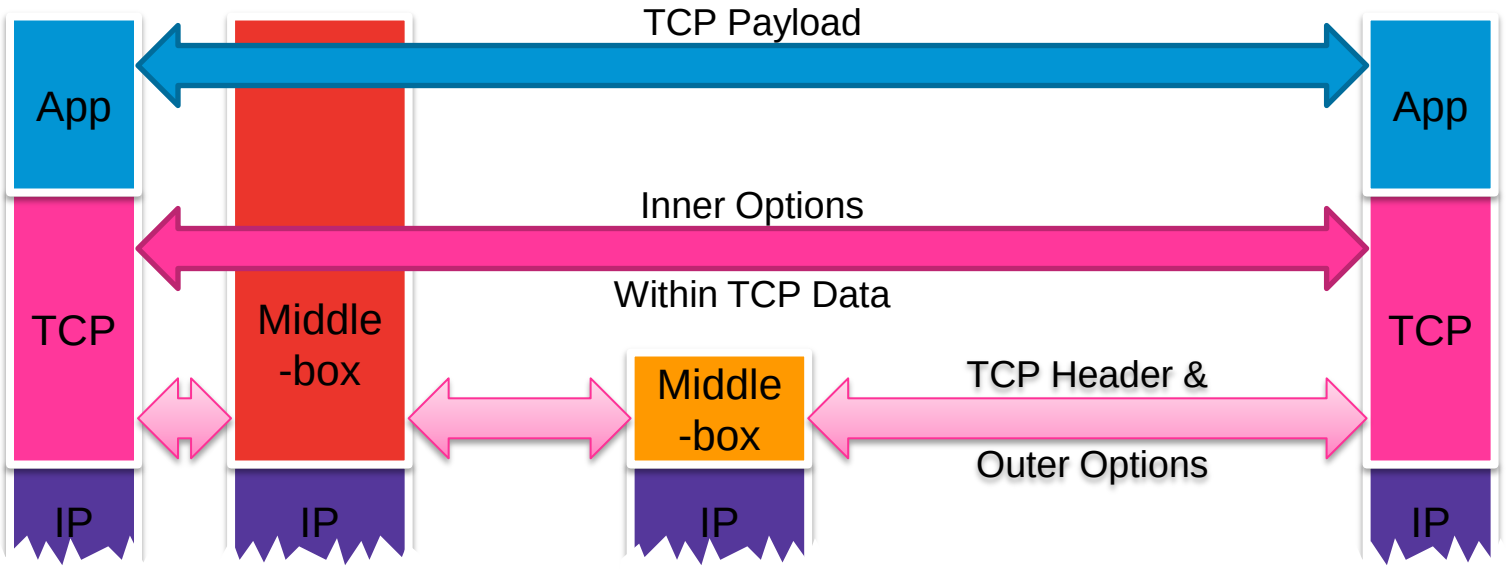
- problem
  hard to extend TCP
    - only 40B for options
    - middleboxes only forward their stereotype of TCP

- purpose of this talk
  introduce proposed solution
  motivate forward thinking (and critical review)

- status
  Extensive Internet Draft posted to IETF TCP mtce & minor mods WG
    - draft-briscoe-tcpm-syn-op-sis -00,01,02
      -> draft-briscoe-tcpm-inner-space-00,-01
    - 5 revisions since Jul IETF, individual draft – no IETF status (yet)
    - triggered by “more space on a SYN is impossible”
      in Joe Touch’s EDO proposal for extra space on non-SYN only (adopted)
    - counter proposal to my other 2 proposals with Joe Touch
      (also individual status)
    - lots of list discussion

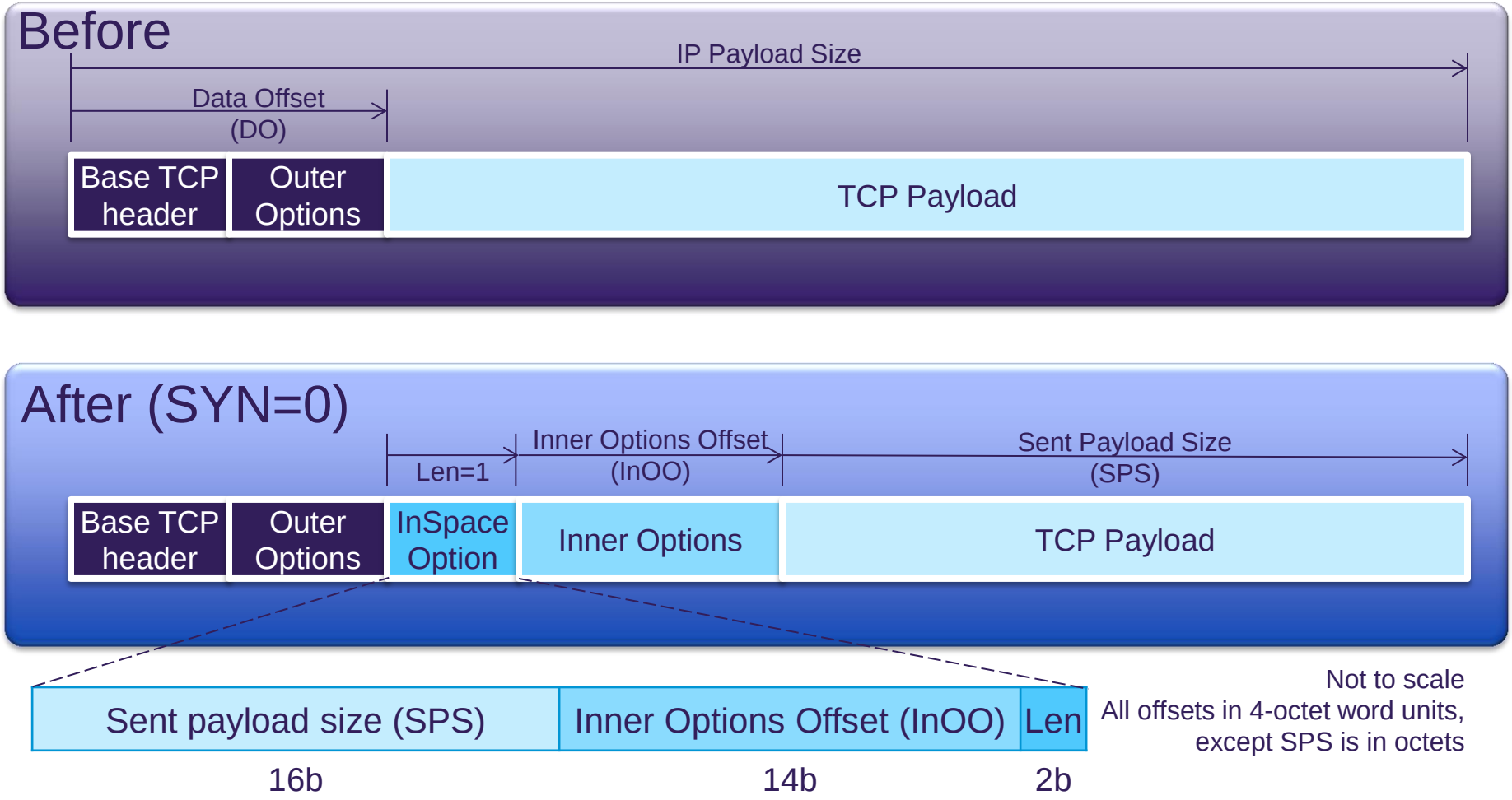
BT

encapsulation model



BT

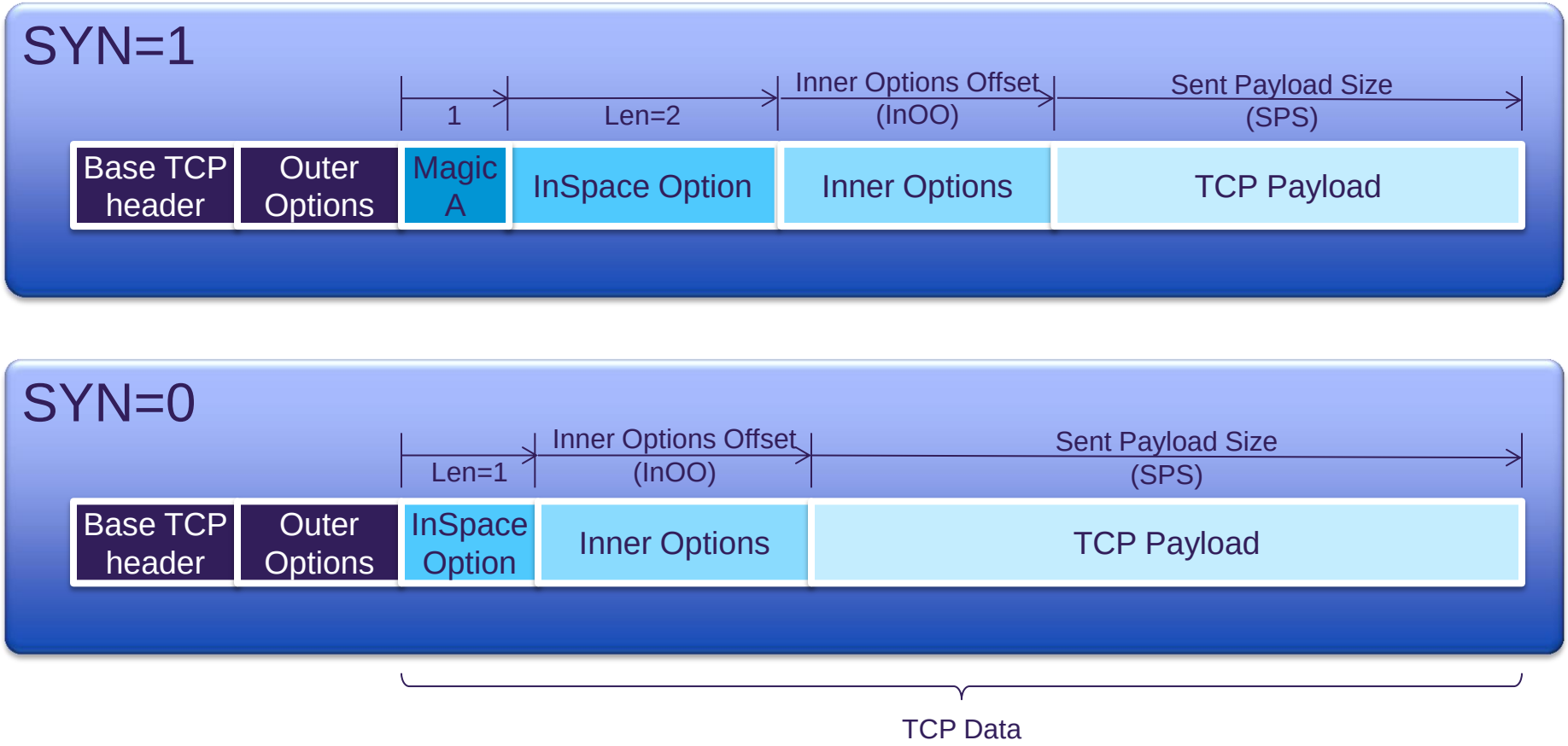
TCP segment structure (SYN=0)



- InSpace solely contains frame size info

BT

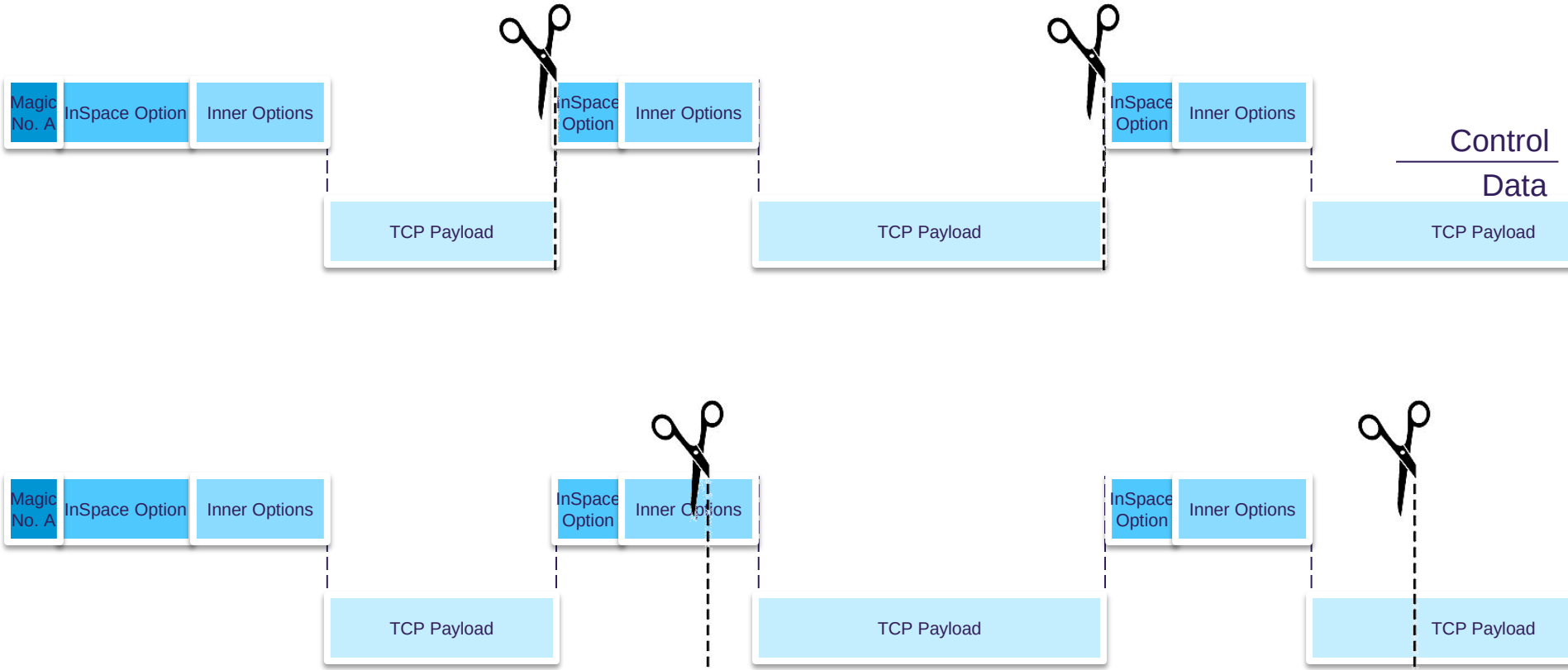
TCP segment structure

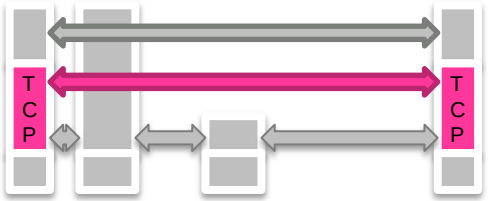


- presence of Inspace flagged by magic no. at start of each stream
- avoided an Outer TCP Option as the flag, which could be stripped
- inherently safe to flag within the payload – shares fate with options

BT

TCP byte-stream





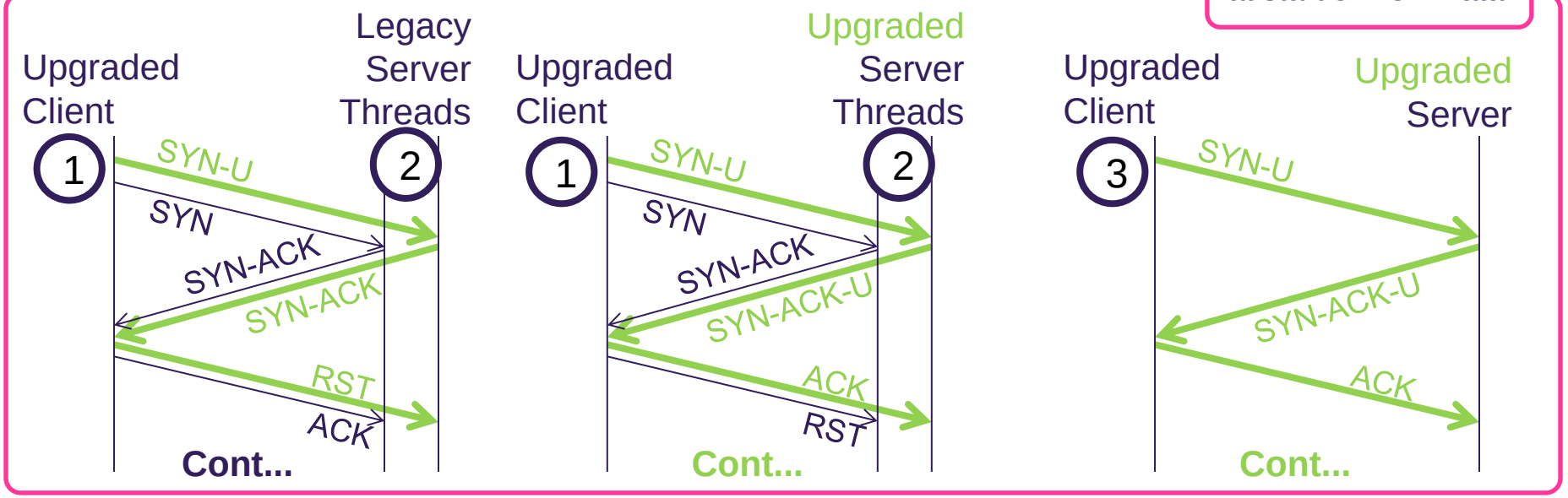
- robust to resegmentation
- Inner Options not prone to stripping
- reliable ordered delivery of Inner Options

BT

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

BT

middlebox domination strategy

non-goal

- evasion of security middleboxes
  - they will evolve to check Inner Options

goal

- Inner Space deployed for something useful (e.g. tcpcrypt)
  - traverses most middleboxes, so reaches critical mass
  - can deploy integrity protection over TCP Data
  - (cannot protect Outer Options today – would fail too often)
- raises stakes
  - then tampering with Inner Options will break comms, not just the theoretical potential benefit of a new option
- puts security middlebox designers on the back foot
  - blocking comms just ‘cos options are non-typical will not sell
  - distinguishing attacks from the latest advances will sell

result

- mess with my options and you shoot yourself in the foot

BT

☺ benefits

1. no arbitary limit on option space
2. incremental deployment
   both of Inner Space itself and of new options it enables
   - middlebox traversal
   - legacy server fall-back
3. no extra handshaking delay
4. reliable ordered delivery of control options

BT

☹ drawbacks - overheads

- Dual Handshake — Example
  - 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}

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

BT

☹ drawbacks - non-deterministic

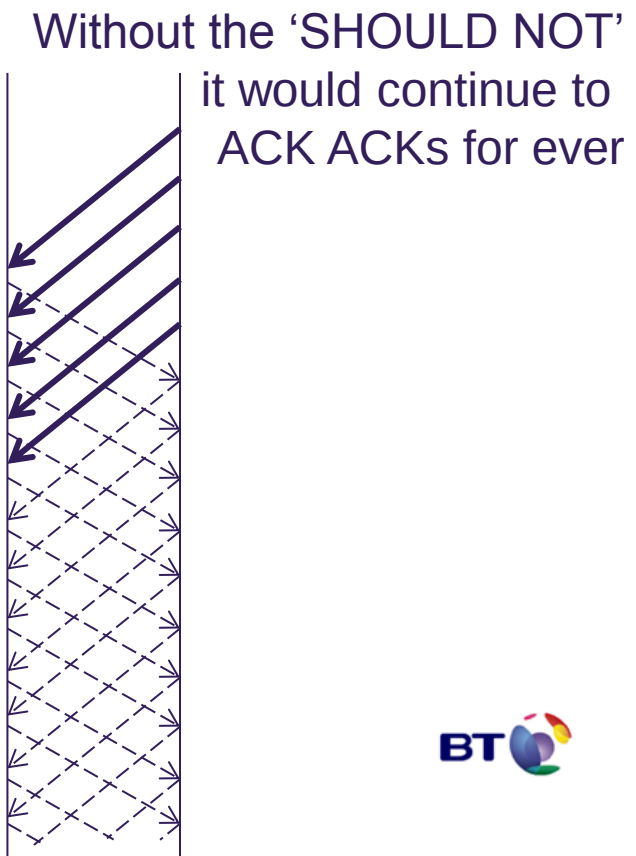
- the magic number approach traverses option stripping middleboxes, but...
- probability that an Upgraded SYN or SYN/ACK is mistaken for an Ordinary Segment: Zero
- probability that an Ordinary SYN or SYN/ACK with zero payload is mistaken for an Upgraded Segment: Zero
- probability that payload data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

BT

tricky bits - zero payload segments

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

- other tricky bits → spare slides or draft
  - option processing order
  - options that alter byte-stream
    - e.g. encrypt or compress
  - the EchoCookie for SYN floods
  - retransmissions during handshake

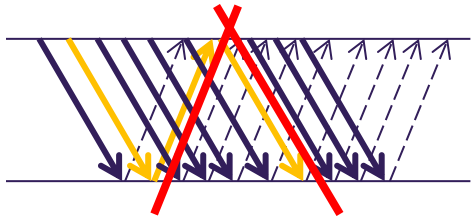


BT

disabling Inner Space temporarily

- set Sent Payload Size (SPS) to special max value 0xFFFF
  - sent segment was not 0xFFFF octets, but behave as if it was
  - values above 0xFFE8 (= 2^{16} – 25) are usable but not believable
- regularly repeat just the 4B InSpace option
  - every 0xFFFF octets (=44.7 * 1466B typical full-sized segments)

- could disable Inner Space for rest of connection
  - separate ModeSwitch TCP option – see spare slide or draft



BT

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

BT

applicability & compatibility (interim*)

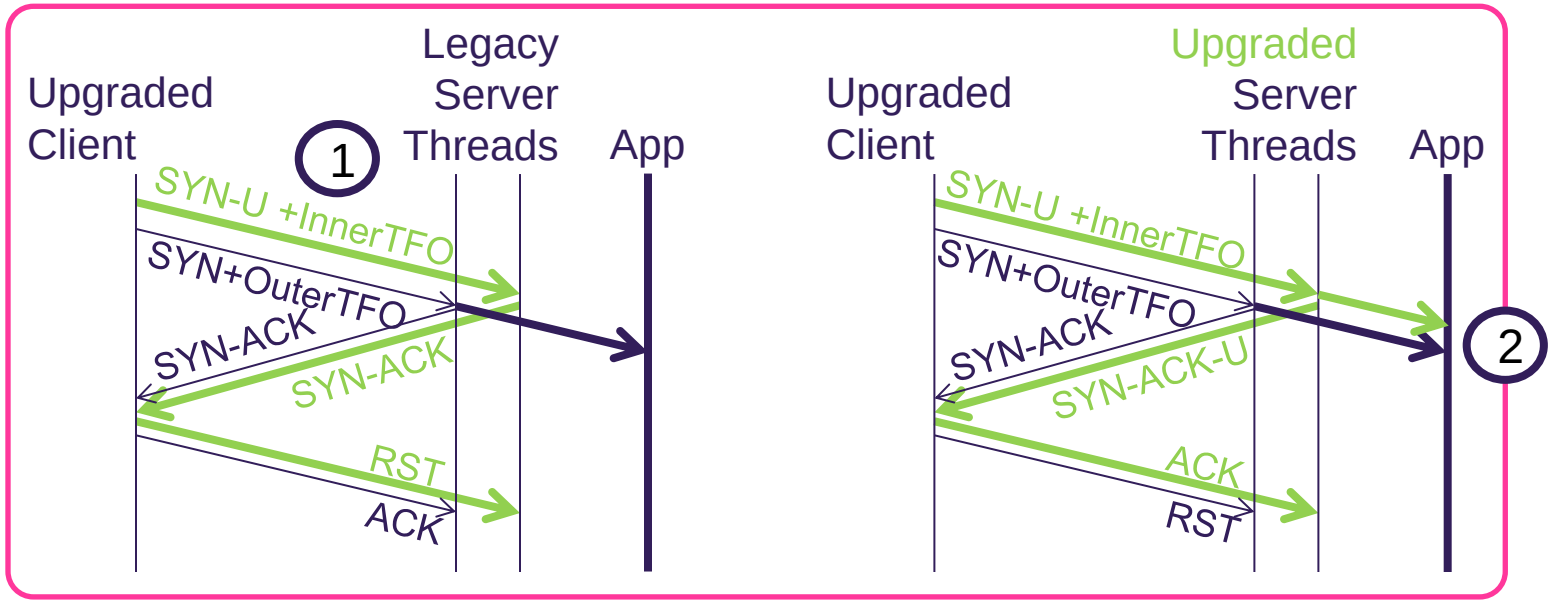
- have to use Outer Option
  typically concerned with individual segments
    - Subsequent Timestamps
    - Selective ACK (SACK)
    - Multipath TCP Data ACK (in Data Sequence Signal – DSS)
- best as Inner Option
  typically concerned data as a stream
    - tcpcrypt CRYPT
- either Inner or Outer Option
  typically starting the connection (and therefore a segment)
    - (4B) Max Segment Size (MSS)
    - (2B) SACK-ok
    - (3B) Window Scale (WS)
    - (10B) First timestamp (TS)
    - (16B) TCP Authentication Option
    - (6-18B) TCP Fast Open (TFO)
    - tcpcrypt MAC
    - (12B) MPTCP except Data ACK

* Many of the above schemes involve multiple different types of TCP option, which need to be separately assessed.

BT

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

BT

Inner Space & tcpcrypt

- tcpcrypt capability negotiation currently adds a round trip
  - **not viable** to add 1RTT delay to every connection to introduce opportunistic encryption

- tcpcrypt currently attempts most of Inner Space
  - in various complicated bespoke ways
  - have proposed how to structure tcpcrypt over Inner Space
  - cuts 1.5 rounds → **makes tcpcrypt viable**
  - cuts out two states – greatly simplifies
  - (?) decouples tcpcrypt from TCP state m/c
  - tcpcrypt can encrypt Inner Options (incl. its own)
    - because that needs reliable ordered delivery

BT

Inner Space & MPTCP

- MPTCP adds Data ACK (in the DSS TCP Option)
  - cumulative ACK of the set of sub-flows cannot be inferred
- Data ACK is a per-segment message
  - cannot use Inner Options
    - would not be interpretable on reception (if out of order)
    - potential deadlock: must not require receive buffer to ACK [1]
- Three ways forward
  1. give up – leave all MPTCP TCP Options as Outer Options
  2. use Inner Space for a low latency MPTCP,
     except DSS and a way to test path for stripping DSS
  3. extend Inner Space to include Outer Options within TCP Data
     without using RWND or sequence space (hard – see next)

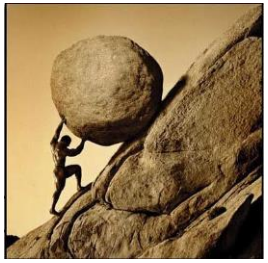
[1] Raiciu et al "How Hard Can It Be? Designing and Implementing a Deployable Multipath TCP"

BT

opportunities / further work

- tcpcrypt-v2 decomposition
- probes
  - any Inner Options delivered reliably in order
- relation to Minion, and multi-stream protocols

- Outer Options in Inner for middlebox traversal



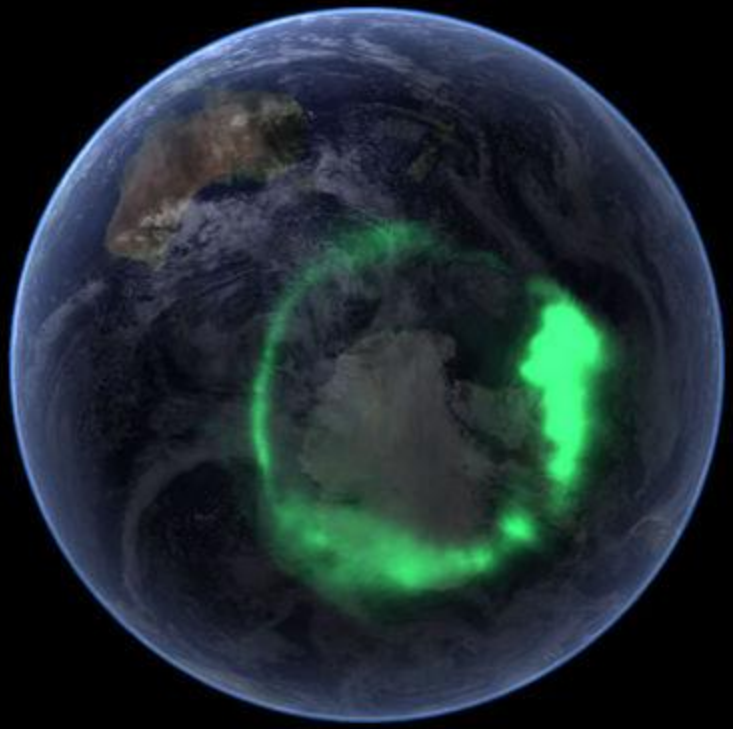
- without consuming rwnd (cf. fixed space for Outer Options)
- without consuming sequence space (avoiding middlebox 'correction')
- delivered immediately in received order, not sent order

BT

next steps

- adoption: IETF TCP mtce & minor mod's
  - comparison with other proposals
  - converge on stable design ASAP
- review focus
  - critique, e.g. design for performance
  - mandatory/optional elements
- path testing
  - is DPI bypass necessary? viable?
- implementation
  - compatibility testing
- IAB workshop on stack evolution in a middlebox Internet

BT

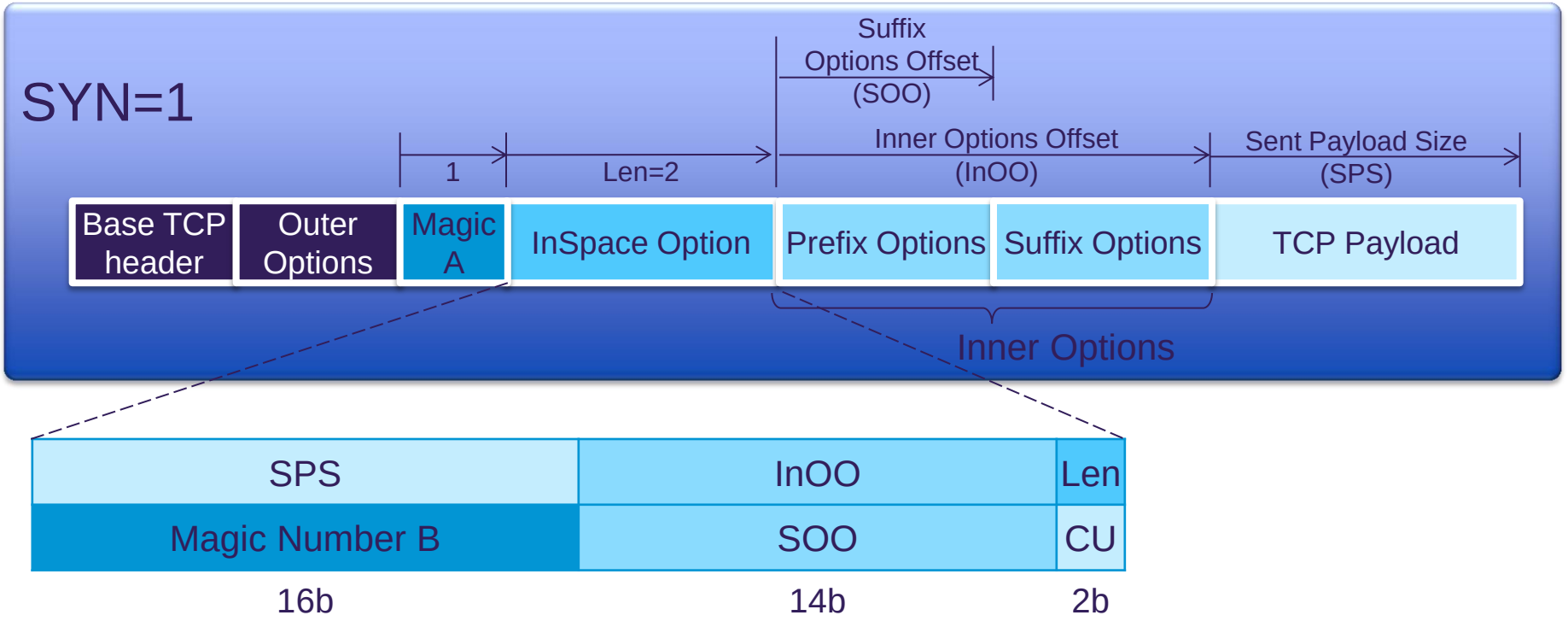


Inner Space

Q&A

Spare slides

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

BT

tricky bits – processing order: one level of recursion

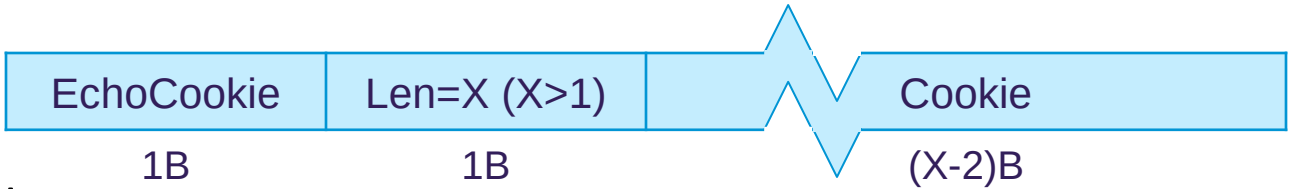
- If TCP alters the TCP Data (e.g. decrypt, decompress in the receiving case, for example)
  - SYN=1: if it hasn't previously found MagicA, it looks again

  - SYN=0: There might be a rekey command in an encrypted Inner Option. So the TCP receiver decrypts up to the end of each set of Inner Options, processes those options, then continues decrypting (which might be with a new key).

BT

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



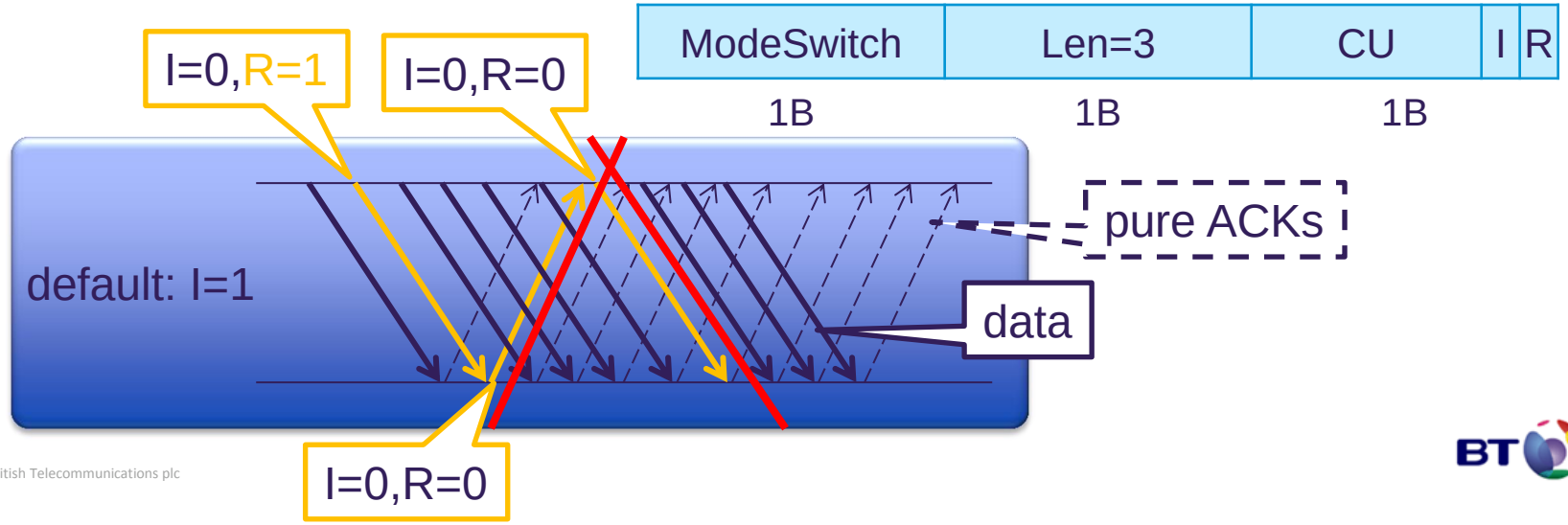
- other opportunities
  - tcpcrypt could use this

BT

extension - ModeSwitch

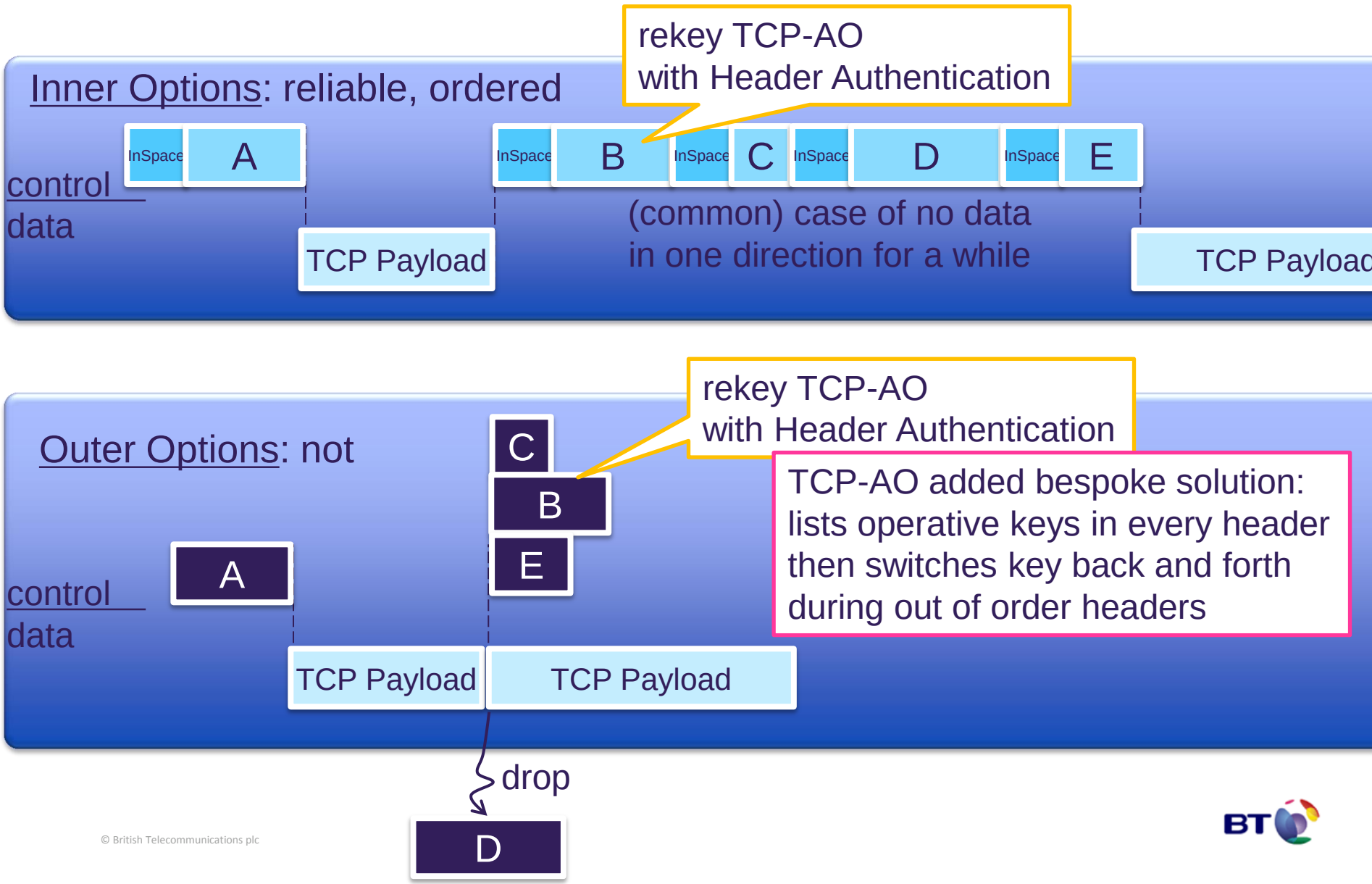
- would be nice to be able to turn off Inner Space protocol
  - each ends need to know when the other end has switched
  - to co-ordinate which will be the last InSpace options
- introduced a generic ModeSwitch Inner TCP option
  - rather than embed an ‘off switch’ into the InSpace option

R : Request mode (special)
I : Inner Space mode
CU : currently unused space for yet-to-be-defined TCP modes



BT

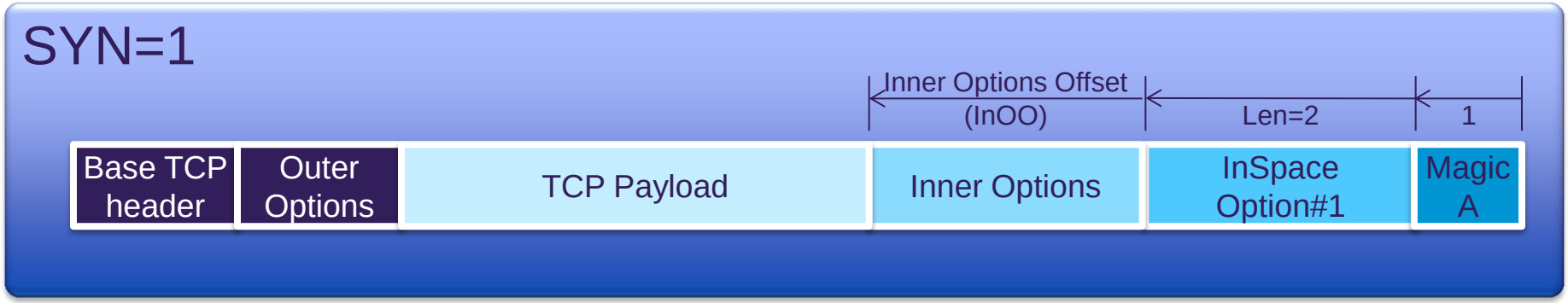
why switching connection mode is tricky

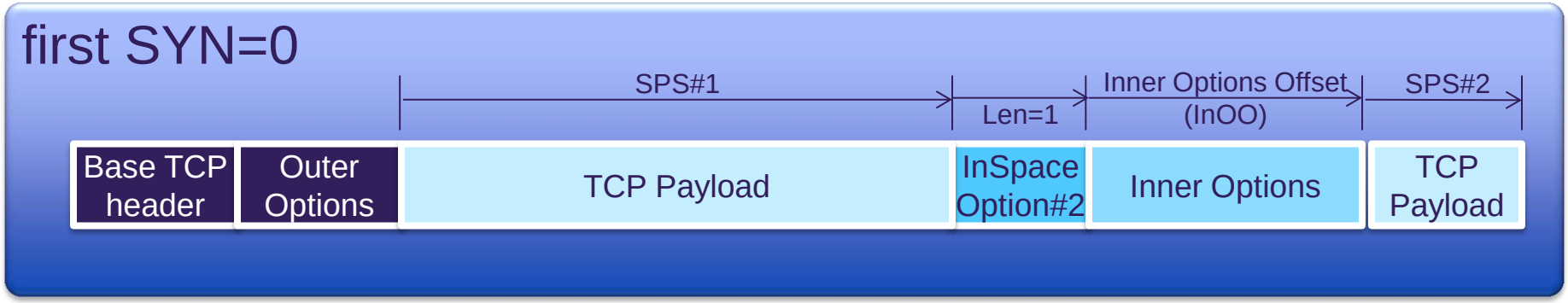


BT

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

BT

spare slides - to write

- handshake retransmissions
- explicit dual handshake
  - corner cases of dual handshake
  - deferred data in SYN

BT