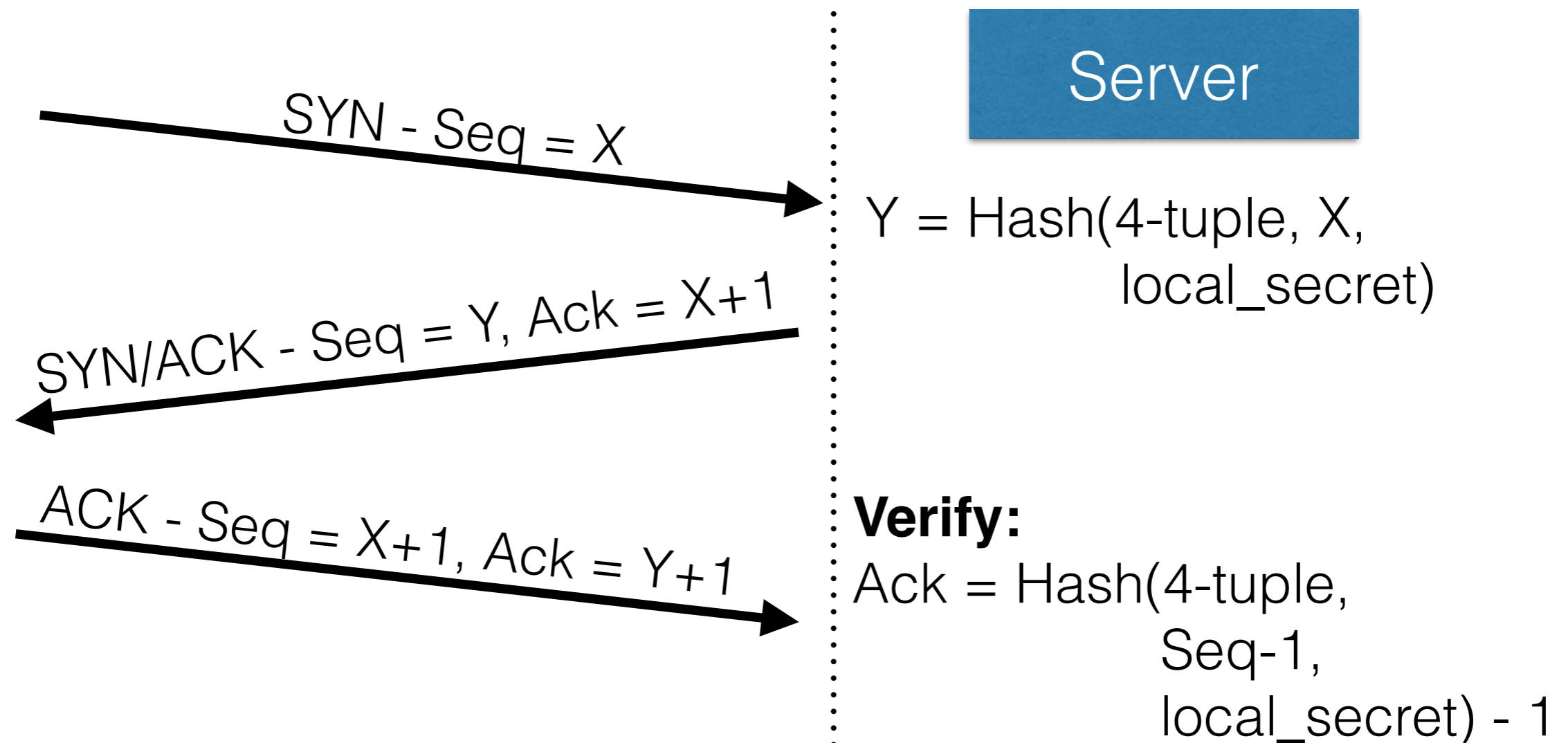# Making Multipath TCP robust for stateless webservers

draft-paasch-mptcp-syncookies
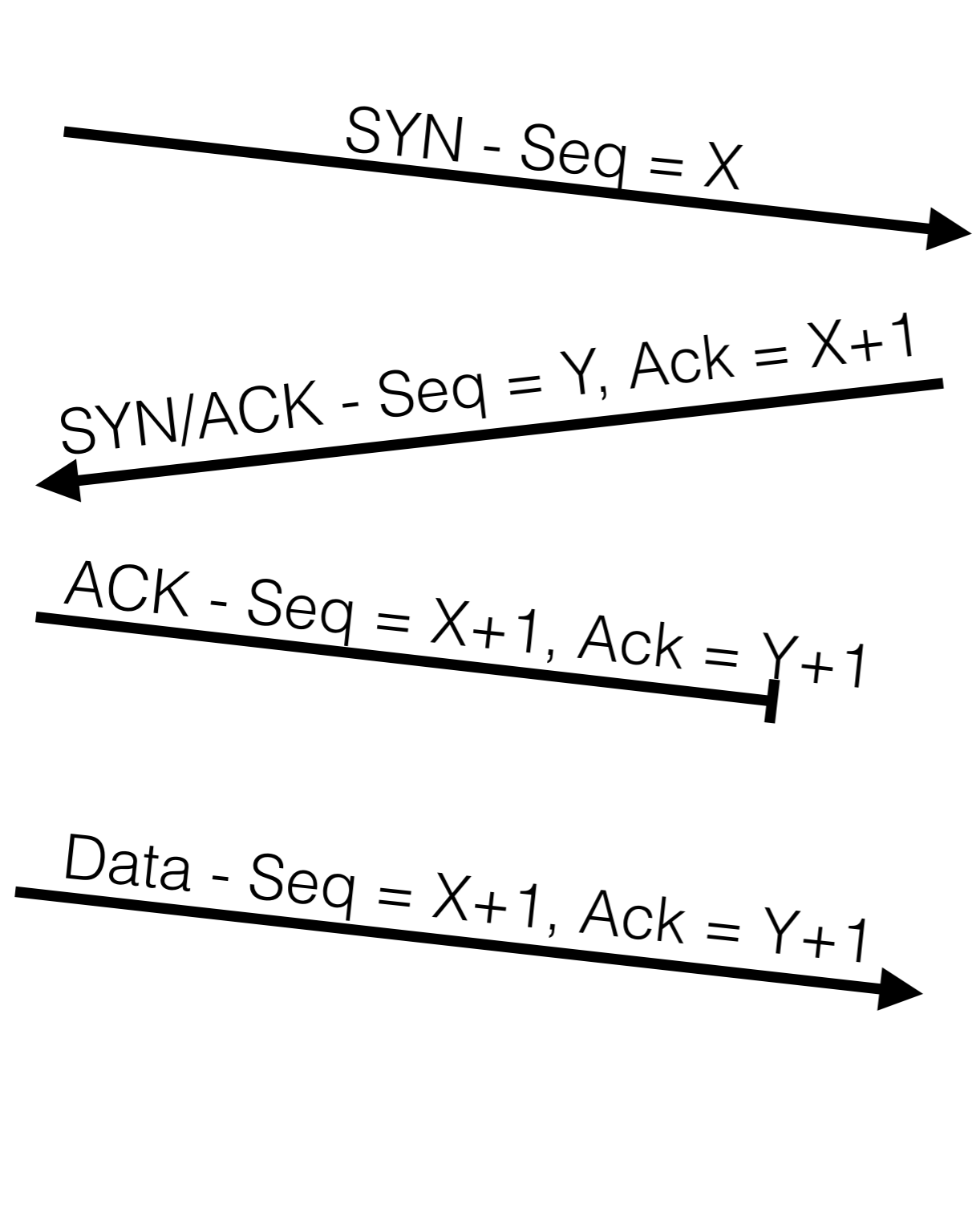
Christoph Paasch <cpaasch@apple.com>
Anumita Biswas <anumita_biswas@apple.com>
Darren Haas <dhaas@apple.com>

TCP SYN-cookies

- Used to handle SYN-flooding attacks

- Server handles SYN in a stateless manner

SYN - Seq = X

Server

$Y = Hash(4\text{-tuple}, X, local\_secret)$

SYN/ACK - Seq = Y, Ack = X+1

ACK - Seq = X+1, Ack = Y+1

Verify:
$Ack = Hash(4\text{-tuple}, Seq\text{-}1, local\_secret) - 1$

Loss of third Ack

Server

SYN - Seq = X

SYN/ACK - Seq = Y, Ack = X+1

ACK - Seq = X+1, Ack = Y+1

Data - Seq = X+1, Ack = Y+1

Verify:
Ack = Hash(4-tuple,
                Seq-1,
                local_secret) - 1

MPTCP with SYN-cookies

- Echoing the keys in the third ACK allows stateless servers for MPTCP

- However, a loss of the third ACK will make MPTCP fallback to regular TCP

    ➡ When packet-loss is high (and MPTCP would provide benefits), there is a higher chance of falling back to regular TCP
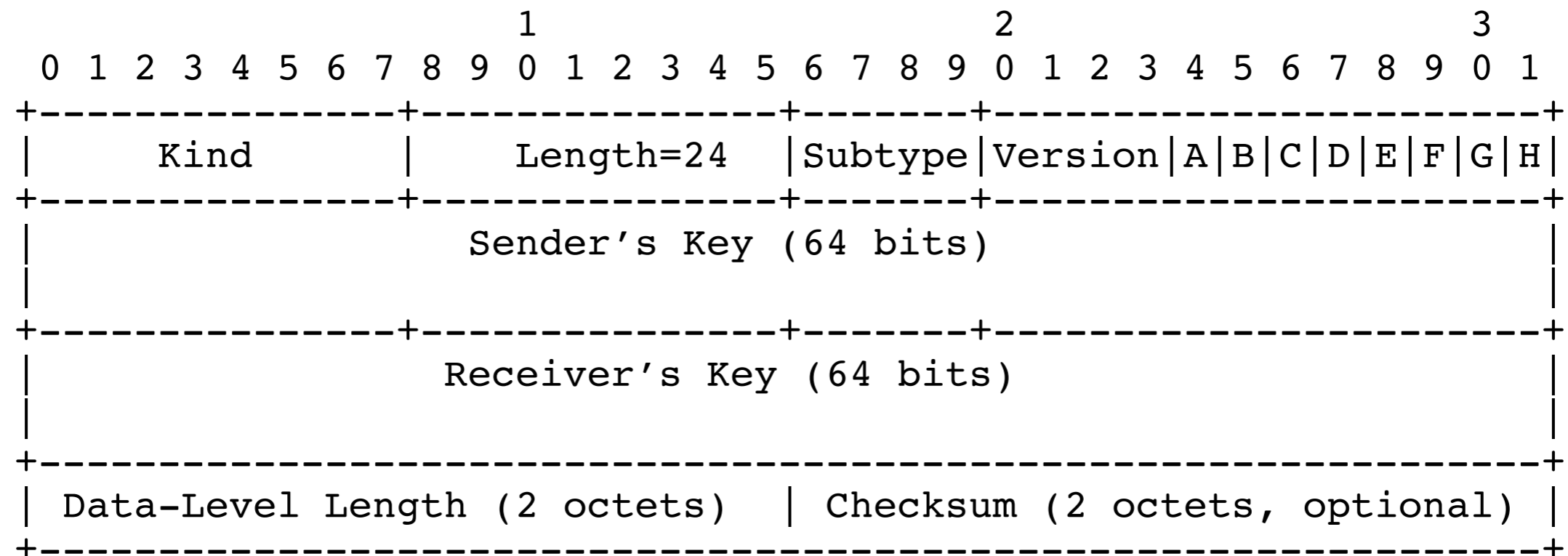
Reliable MP_CAPABLE

- Make MP_CAPABLE reliable
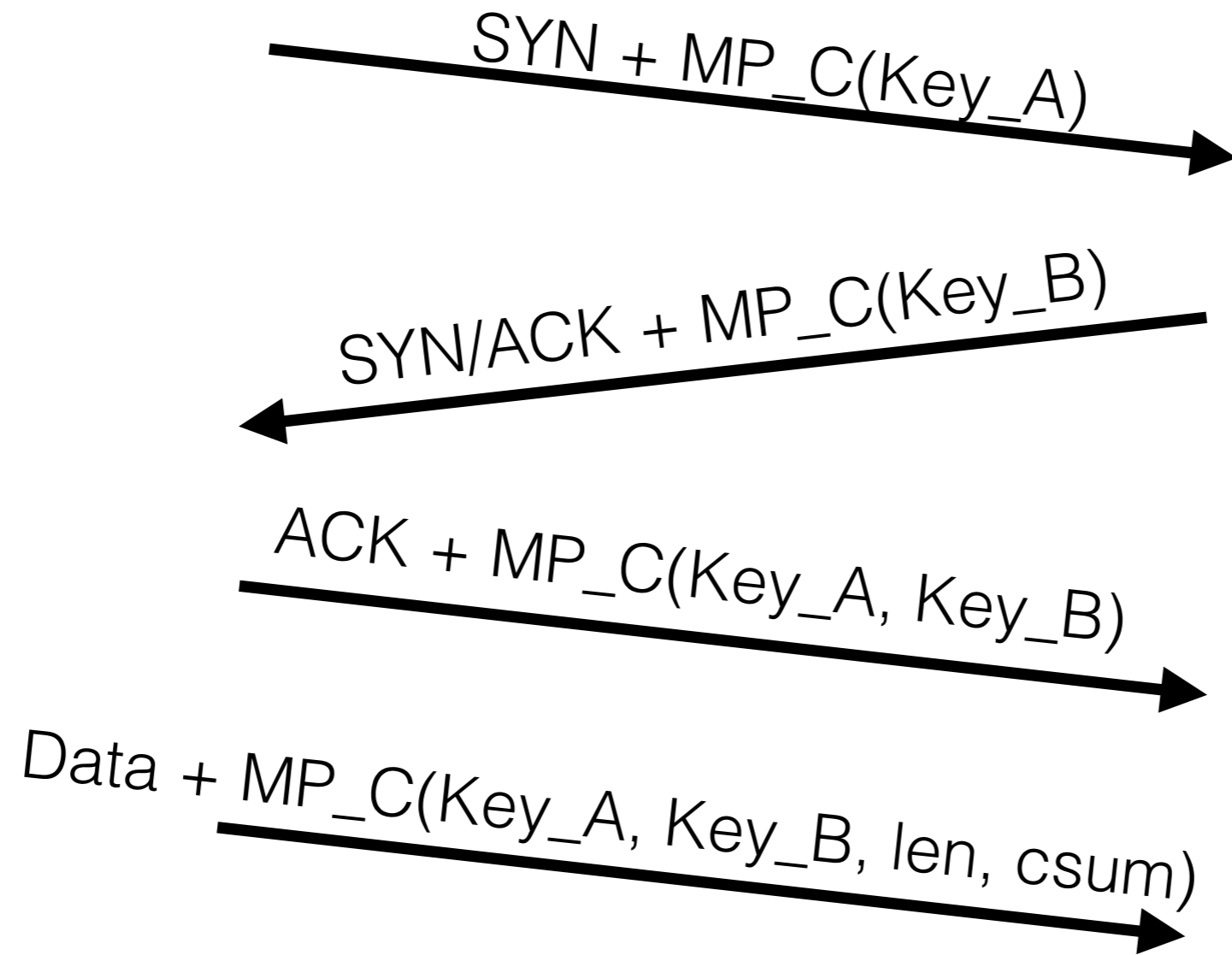
- Combining data with MP_CAPABLE allows for reliability

SYN + MP_C

SYN/ACK + MP_C

ACK + MP_C
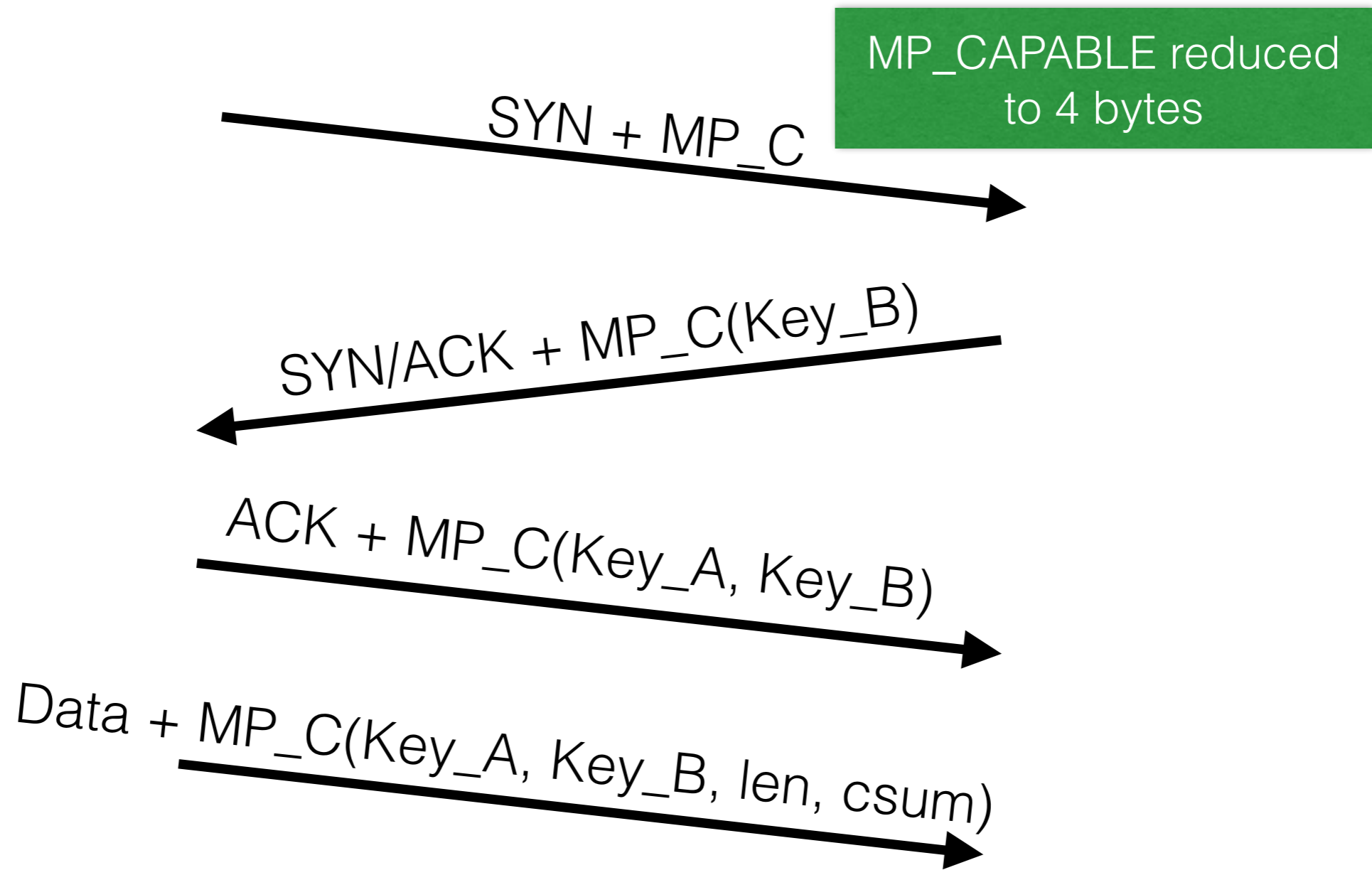
Data + DSS + MP_C(Key_A, Key_B)

Running out of
TCP-option space
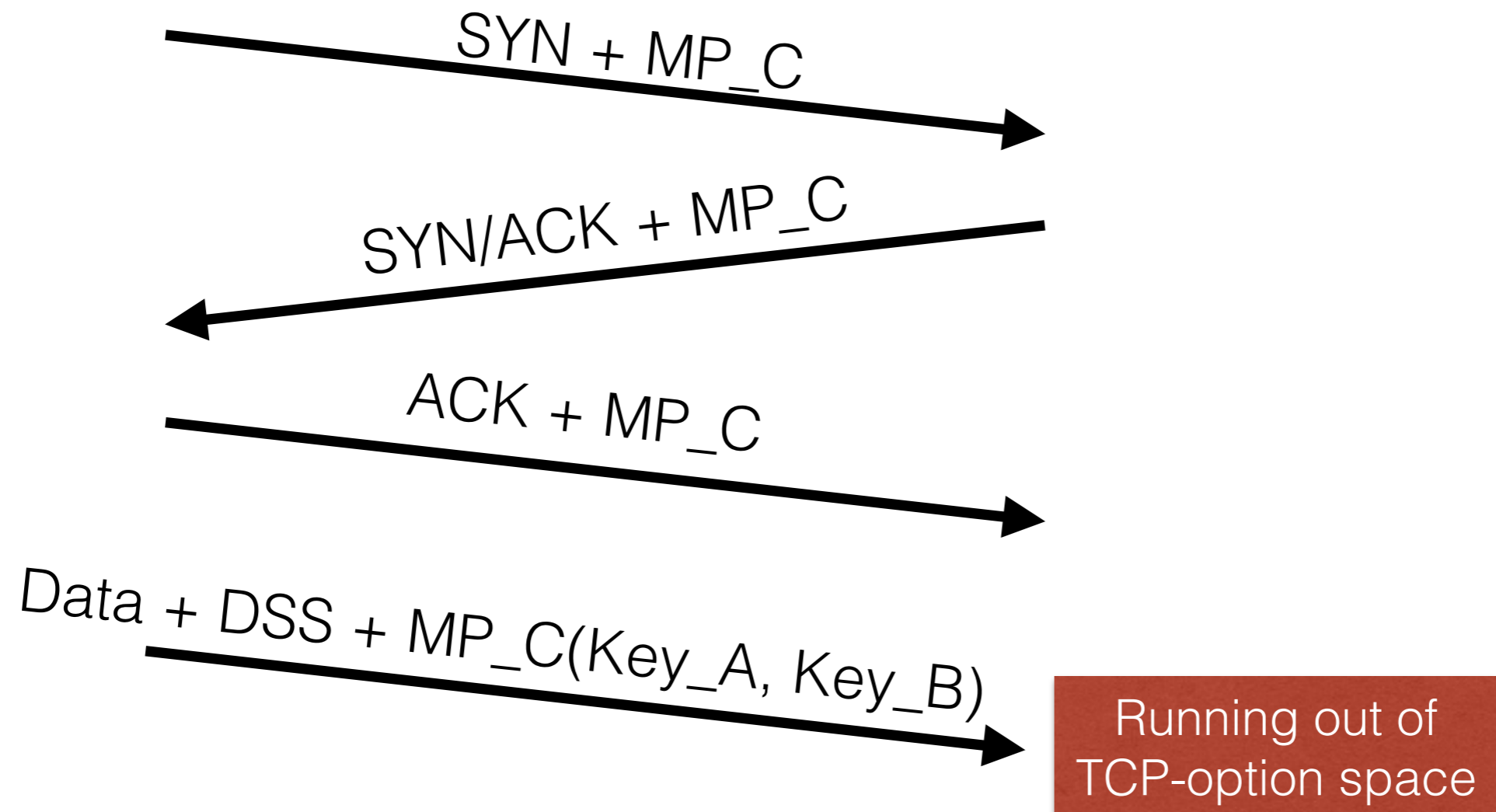
Combining MP_CAPABLE with DSS-option

- Only sent on the first data-segment

  ➡ data-sequence number = IDSN

  ➡ subflow-sequence number = 1

```
                    1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+-------+----------------------+
|     Kind      |  Length=24    |Subtype|Version|A|B|C|D|E|F|G|H|
+---------------+---------------+-------+----------------------+
|                    Sender's Key (64 bits)                    |
|                                                              |
+---------------+---------------+-------+----------------------+
|                   Receiver's Key (64 bits)                   |
|                                                              |
+--------------------------------+-----------------------------+
| Data-Level Length (2 octets)   | Checksum (2 octets, optional)|
+--------------------------------+-----------------------------+
```

Reliable MP_CAPABLE

SYN + MP_C(Key_A)

SYN/ACK + MP_C(Key_B)

ACK + MP_C(Key_A, Key_B)

Data + MP_C(Key_A, Key_B, len, csum)

Gaining option-space in the SYN

SYN + MP_C

MP_CAPABLE reduced to 4 bytes

SYN/ACK + MP_C(Key_B)

ACK + MP_C(Key_A, Key_B)

Data + MP_C(Key_A, Key_B, len, csum)
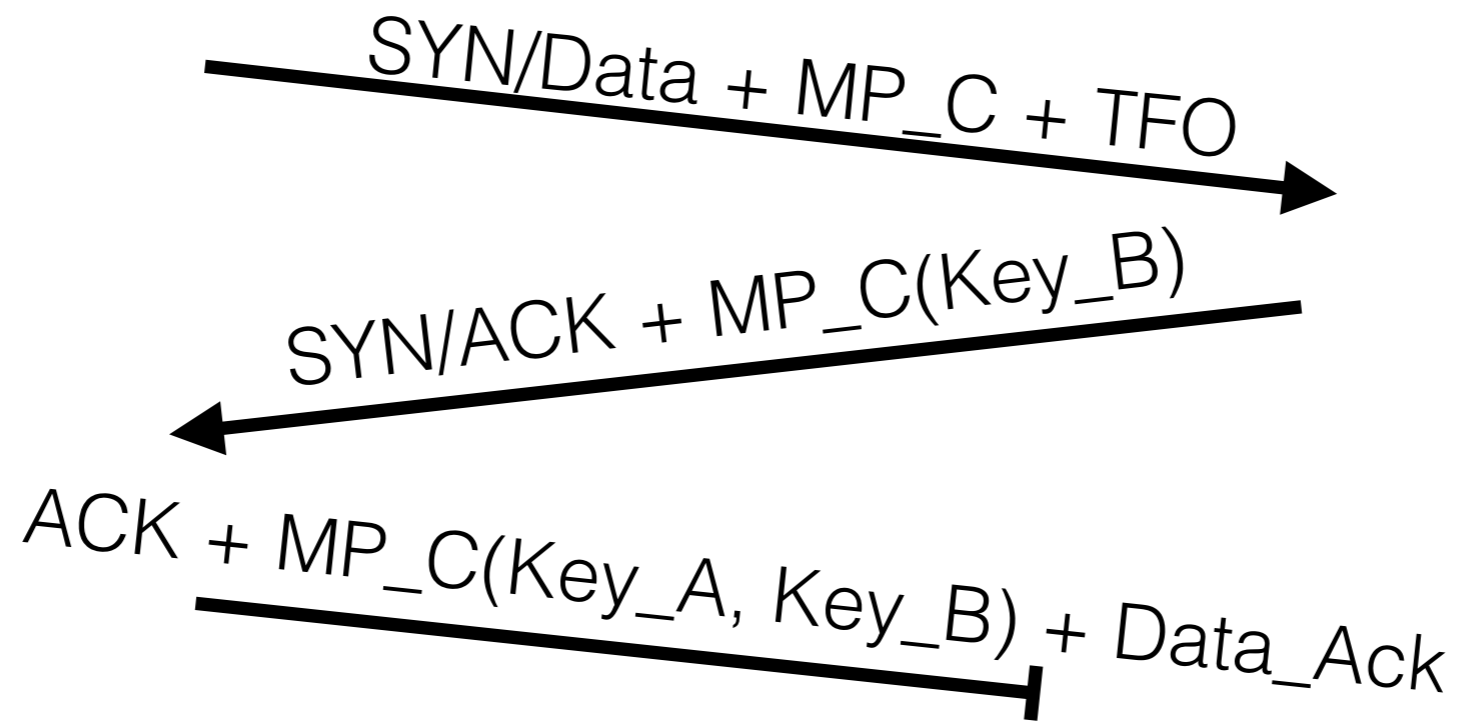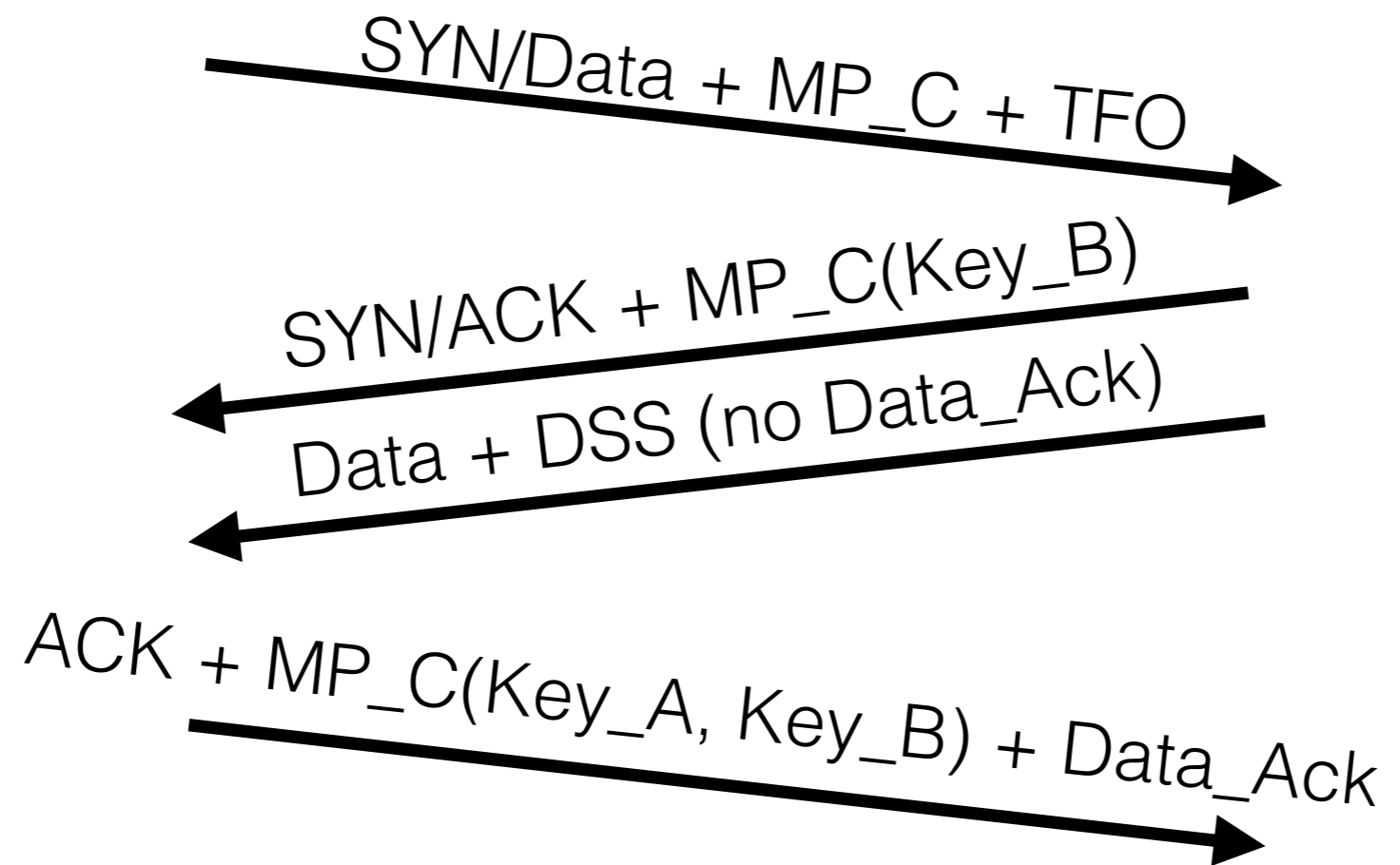
Reliable MP_CAPABLE

- Fully enables stateless web servers

- opens the door to reduce MP_CAPABLE in the SYN down to 4 bytes

  ➡ Need to ensure delivery of MP_CAPABLE in case of TFO (details are in the draft)

- Inclusion into RFC6824bis ?

Backup (MPTCP + TFO)

SYN/Data + MP_C + TFO

SYN/ACK + MP_C(Key_B)

ACK + MP_C(Key_A, Key_B) + Data_Ack

Backup (MPTCP + TFO)

SYN/Data + MP_C + TFO →

← SYN/ACK + MP_C(Key_B)

← Data + DSS (no Data_Ack)

ACK + MP_C(Key_A, Key_B) + Data_Ack →

- Server must reply to ACK + MP_CAPABLE with Data_Ack

  - Acknowledge reception of MP_CAPABLE

- Client must send MP_CAPABLE + Data_Ack until it received one Data_Ack