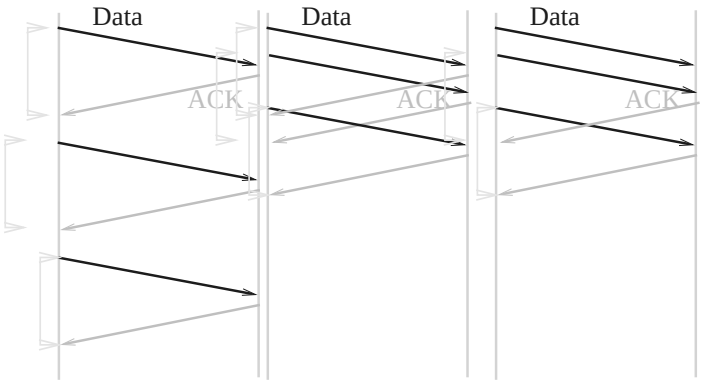
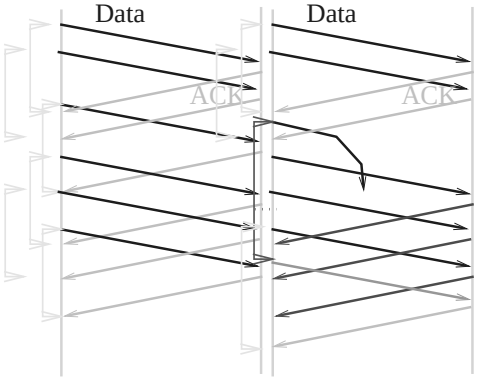
Round Trip Time Estimation

- Useful to know when to retransmit
  - if have not received ACK within the RTT
    - (plus a bit)
    - then assume packet lost
- But how to measure the RTT?
  - Measure delay between packet and its ACK
    - easy
  - But
    - Send packet
      - wait ... wait ... wait (nothing)
    - Retransmit packet
    - ACK arrives
  - Which packet was acknowledged?
    - The initial packet
      - acknowledged slower than expected
    - Or the retransmit?

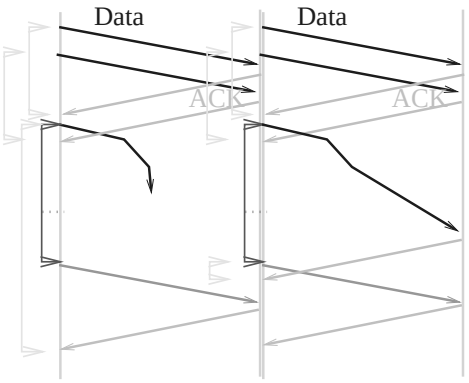
TCP RTT Measurement



TCP RTT Measurement (2)



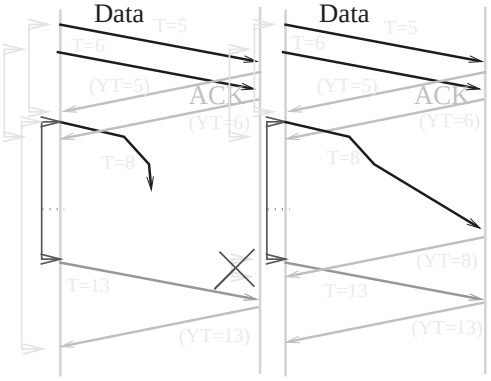
TCP RTT Measurement (3)



TCP Timestamp Option

- Each TCP can add timestamp option to every packet
- Peer TCP sends back timestamp received with each ACK
- Allows TCP to determine which packet was ACK'd
- Better than that
  - no need to remember when packets were sent
  - returning timestamp contains that information
- Also used for long delayed old packet detection
  - extends the sequence number space

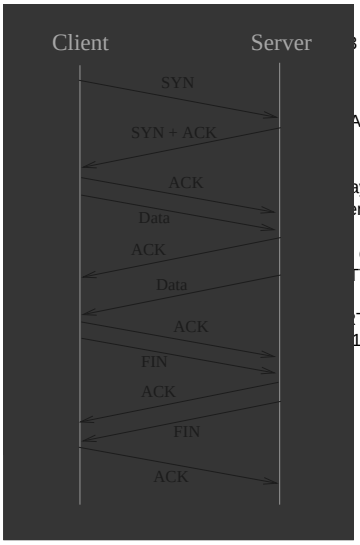
TCP RTT Measurement (tstamp)



TCP or UDP

- UDP is unreliable, not flow controlled
- TCP is reliable, has flow control
  - Often would prefer to use TCP to UDP
- But
  - Overheads are much greater

Typical TCP



way handshake

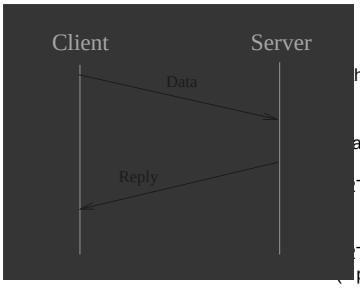
A

ay before reply
er RTT

connection
TT (but unimportant)

TT + server delay
1 packets)

UDP Alternative



he data

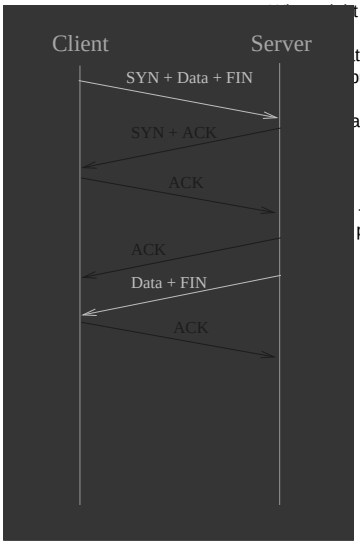
ay

TT

TT + server delay
packets)

- 2 packets instead of 11
- 1 RTT instead of 2

Minimal TCP



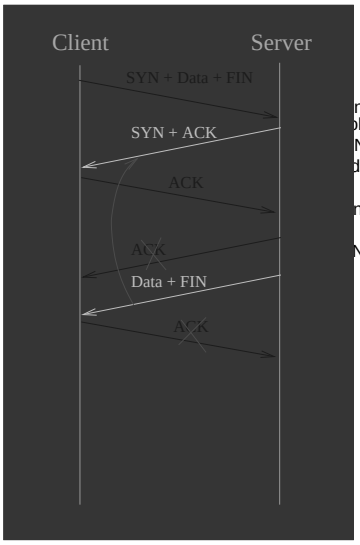
t be merged in TCP?

ta in SYN packet
ut FIN in SYN packet

a reply can carry FIN

+ server delay
packets)

TCP Cannot Do

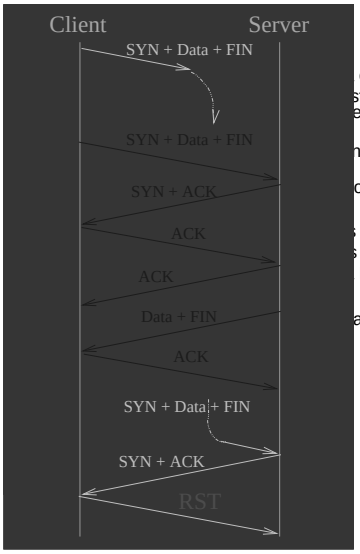


not return data with its SYN
hibits passing any data in
N to application until
dshake is complete

not known that SYN is new

N+ACK acks only the SYN

Old Duplicate SYN



client sending a SYN
st in the net and
ed

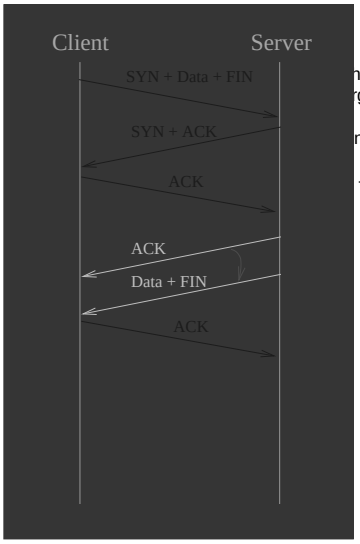
n continues as normal

ost" SYN arrives

normal SYN+ACK reply
not expecting this
RST

a were already delivered?

TCP Can Sometimes Do



n ACKs data in SYN packet)
rged with DATA reply packet

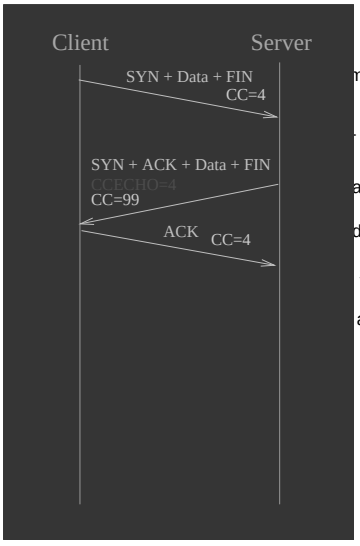
n Server Delay

+ server delay
(5 packets)

Maximum Transaction Rate

- Bounded by 2 * RTT + server delay
- But also limited by TIME WAIT state
  - No more data on same connection for 2*MSL
  - Client picks a different port number
    - different connection
- If 1000 connections / second, and MSL == 2 mins (120 secs)
  - Then 240*1000 connections in TIME WAIT state
    - Consumes lots of memory
    - (if 40 bytes/connection, almost 10MB)
- Worse! Impossible, only 65536 ports!
  - Thus limited to about 270 trans/sec

T/TCP Packet exchange



mal T/TCP 3 packet
transaction

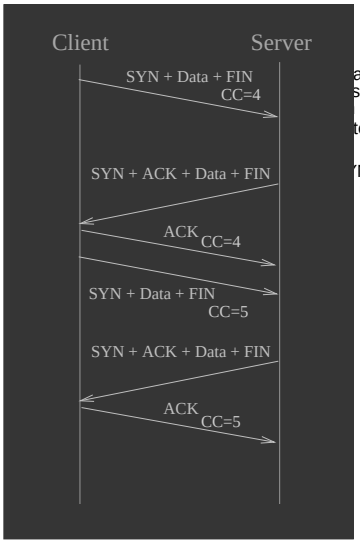
Server Delay

added to SYN
old segment protection
d in all packets

SYN has its own CC

also carries echo of
received CC from SYN

T/TCP transactions



arnation of connection
s soon as desired after old
ends - the CC option allows
tes to be detected.

'N also serves (if needed) as
FIN

T/TCP Server choices

- Server can perform 3 way handshake whenever it needs to
  - Received lower CC than expected
  - Didn't know what CC to expect
  - ...
  - (And if doesn't implement T/TCP)

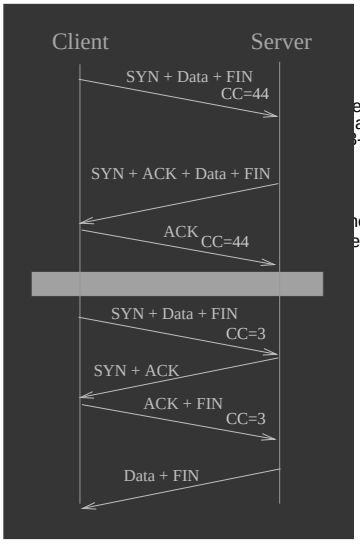
T/TCP Client choices

- Client needs to be able to force 3 way handshake
  - Could just omit CC option
    - But that tells server that T/TCP is not supported
  - Could deliberately send lower CC
    - Hard to know what is low enough, not too low
  - Could just not send data in SYN packet
    - That could mean slightly increased delays

T/TCP Client choices (2)

- Also need a way to resynchronise server with client's CC
- New option CCnew added
  - Replaces CC option in initial SYN packet (only)
  - Forces 3-way handshake before connection established
    - (before data handed to server application)
  - Informs server of clients CC value
    - once 3 way handshake is complete

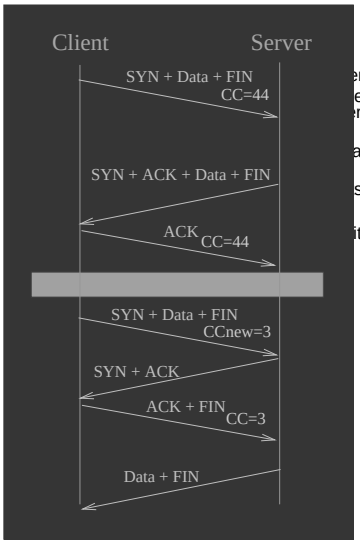
T/TCP Client Lost State



es state
a lower CC number
-way handshake

new connection until
r than server's memory

T/TCP CCnew

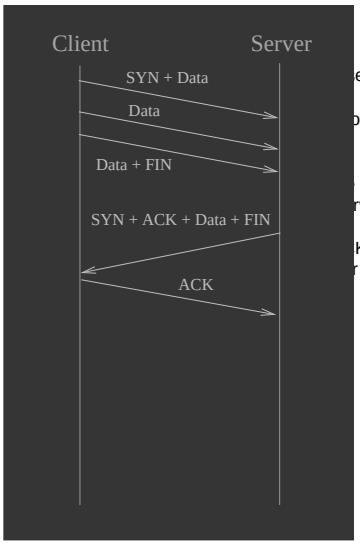


nt sends a new option
enever it has no
rver

ays does 3 way handshake
Verifies CCnew is valid
s memory to CCnew value

itial SYN packet uses CCnew

T/TCP Large Requests



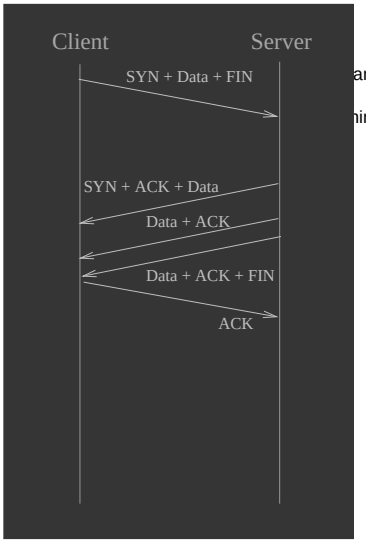
e T/TCP for large requests

othing new from TCP here

of initial packets from
rver carry ACK

K anything until ISN received
r in its SYN+ACK packet

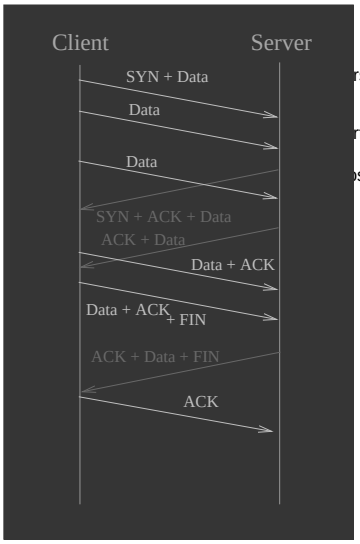
T/TCP Large Replies



andle large replies

hing different from TCP

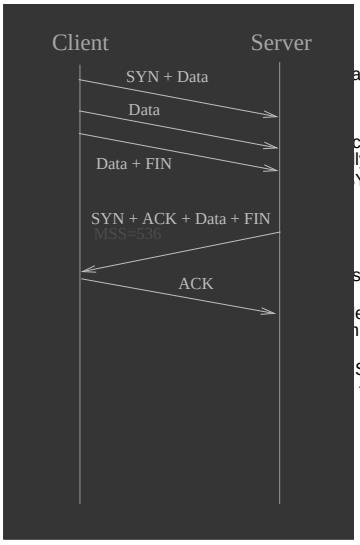
T/TCP Large Request & Reply



rse,
both

rt from CC options
(not shown)
st just the same as TCP

T/TCP Large Requests Revisited



a problem here?

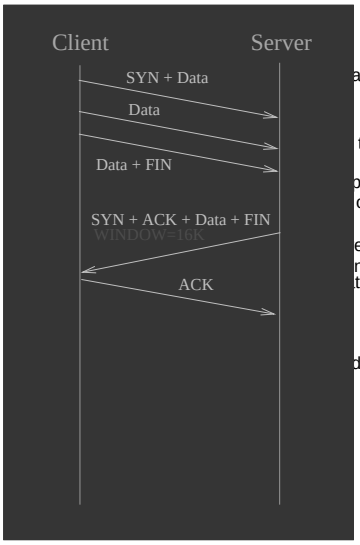
client know the MSS?
ly transmitted with
YN

server info cached

es server's CC
n PMTUD to server

SS from previous
to cache

T/TCP Large Requests Again



another problem here?

the window size

packets
only to one connection

eceived no packets
nection when it is
ta

dow before known = 4K