A Test To Allow TCP Senders to Identify Receiver Cheating

Toby Moncaster, Bob Briscoe, Arnaud Jacquet
BT PLC

draft-moncaster-tcpm-rcv-cheat-00.txt

Intended for Standards Track

Background

- TCP senders rely on accurate feedback from their receivers about congestion
- A dishonest receiver can:
  - optimistic acks: fool a sender into increasing its rate
  - conceal lost data segments: to avoid a congestion response
- These attacks disrupt sharing of sender's own resource
- Both these attacks can fool sender into using excess network resources by concealing the presence of congestion
- There are some existing proposed solutions:
  - Randomly skipped segments – hold back a segment and transmit it once a duplicate acknowledgement is received
  - The ECN nonce – the receiver has to guess the correct nonce sum (NS) [RFC3540] for any lost segment or optimistic ack
  - A transport layer nonce – add a specific nonce to the transport layer which has to be echoed back to the sender

Requirements for a Robust Solution

In order to deal with such dishonesty, the sender has to identify that it is occurring. This can be done using some form of testing of the receiver. Any such test should meet certain requirements:

1. Any test mustn’t adversely affect existing congestion control and avoidance algorithms
2. Any test should utilise existing features of the TCP protocol
3. It shouldn’t require the use of any negotiable TCP options
4. The receiver shouldn’t play an active role in the process
5. If this is a periodic test, the receiver mustn’t be aware that it is being tested for honesty
6. If the sender actively sanctions any dishonesty it identifies, it should be certain of the receiver's dishonesty before taking action against it
7. The test shouldn’t harm an innocent receiver

Assessing proposed solutions

	Skipped Segments	ECN nonce	TCP nonce
Doesn't interfere with congestion control	✓	✓	✓
Utilise existing features	✓	x	x
Receiver plays passive role	✓	x	x
No negotiable TCP options	x	x	x
Receiver unaware of testing	✓	n/a	n/a
Able to prove dishonesty?	✓	✓	✓
Doesn't harm innocent receiver	x	✓	✓

Our proposed Solution

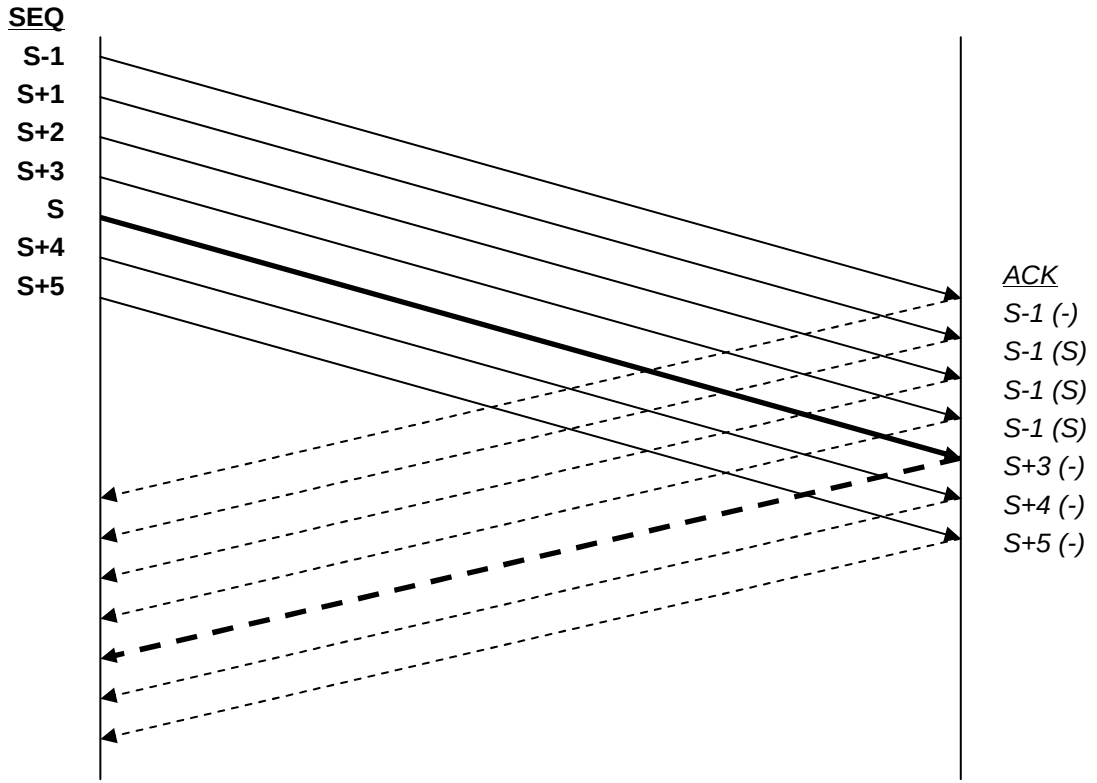
Our proposed solution is based on Rob Sherwood's Randomly Skipped Segments solution. But we avoid harming innocent receivers by adding an initial stage

Key goal was to design system using basic TCP behaviour of receivers. Approach allows honest senders to identify dishonest receivers but not harm innocent receivers…

1. Delay a segment by a small amount to trigger duplicate acks. If a receiver doesn't send these correctly then move to stage 2.
2. Delay a segment until a duplicate ack is received showing the receiver is aware of the gap.

Stage 1 Test

- To test a receiver, select segment S and displacement D where 2 < D < K-2, K = current window size. Segment S is transmitted after segment S+D
- Receiver should generate D duplicate acks



Assessing Stage 1 of the New Test

Comparing the stage 1 test against the list of requirements:

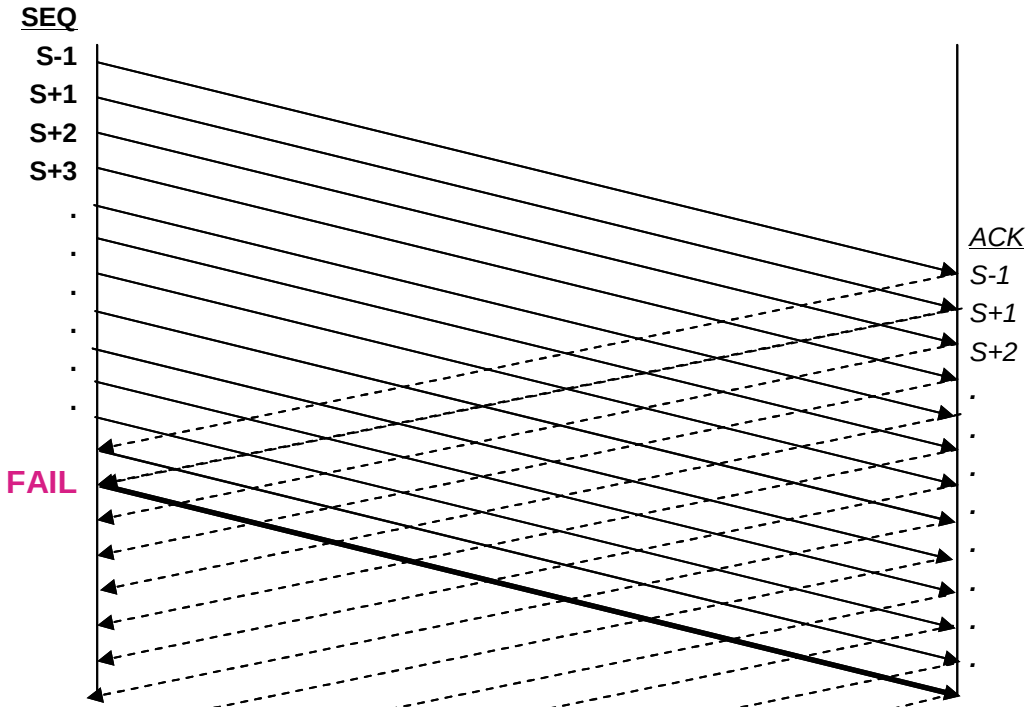
Doesn’t interfere in congestion control	✓
Utilise existing features	✓
Receiver plays passive role	✓
No negotiable TCP options	✓
Receiver unaware of testing	✓
Able to prove dishonesty?	*
Doesn’t harm innocent receiver	✓

* The test doesn’t prove dishonesty but failing it strongly suggests dishonesty

Stage 2 Test

If a receiver fails the first test we can perform a tougher test similar to Sherwood's skipped segment test

- Select a segment, S. Start a congestion response. Don't transmit S until you receive a dup. ACK for segment S-1
- If you receive an ACK for a segment between S-1 and R, then the receiver fails the test



Assessing Stage 2 of the New Test

Comparing the stage 2 test against the list of requirements:

Doesn’t interfere in congestion control	✓
Utilise existing features	✓
Receiver plays passive role	✓
No negotiable TCP options	✓
Receiver unaware of testing	✓
Able to prove dishonesty?	✓
Doesn’t harm innocent receiver	*

* By this stage we are already suspicious of the receiver

Conclusions

- Receiver cheating can skew a sender's allocation of its own resources (getting the cheating receiver more resources)
- Receiver cheating can fool the sender into giving excess network resources to the receiver, possibly causing collapse
- This 2-stage test allows senders to easily identify cheating receivers
- It causes minimal problems for honest receivers
- It encourages honest behaviour because cheating carries delay penalty
- Only requires modification of sender behaviour so easy to implement
- If enough senders implement this it provides protection to network

draft-moncaster-tcpm-rcv-cheat-00.txt

Questions?