Dealing with sequence-number randomizing firewalls

*Benjamin Hesmans, **Christoph Paasch**,*
Olivier Bonaventure

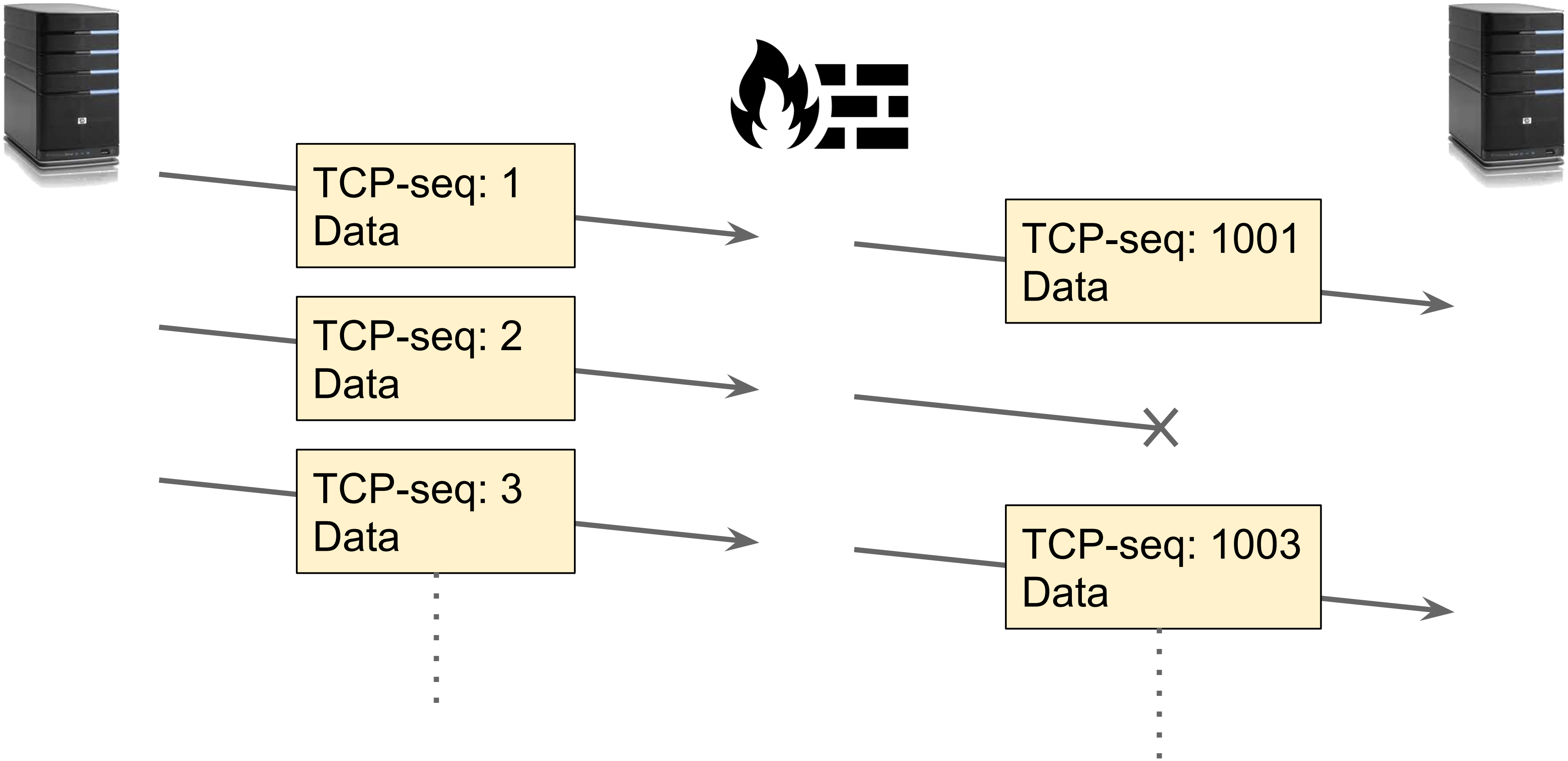
UCLouvain, IP Networking Lab

Sequence number randomization

- Some (old) TCP-stacks did not sufficiently randomize their initial TCP sequence number
- Firewalls "fixed" it by randomizing TCP sequence numbers

But some of them forgot SACK

Sequence number randomization



TCP-seq: 1
Data

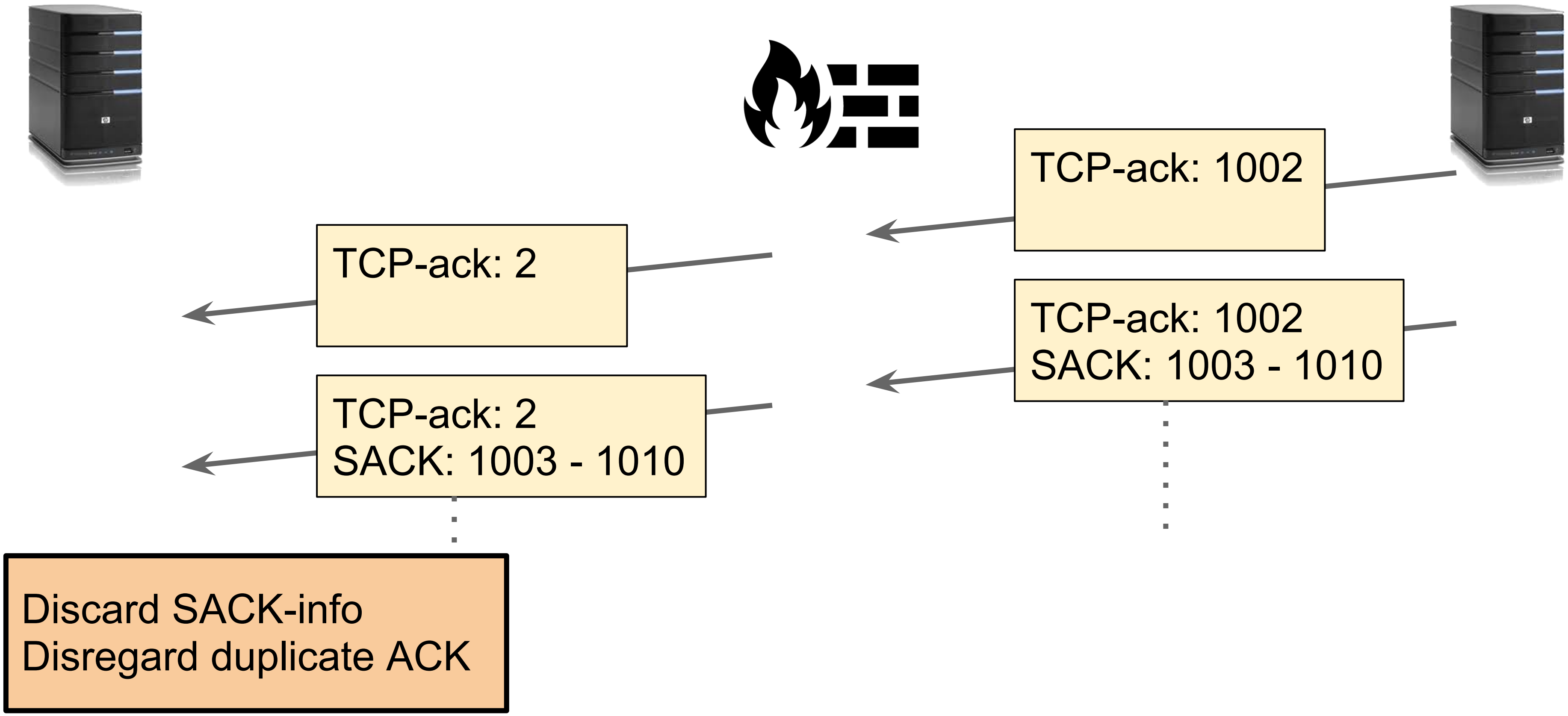
TCP-seq: 2
Data

TCP-seq: 3
Data

TCP-seq: 1001
Data

TCP-seq: 1003
Data

Sequence number randomization



TCP-ack: 1002

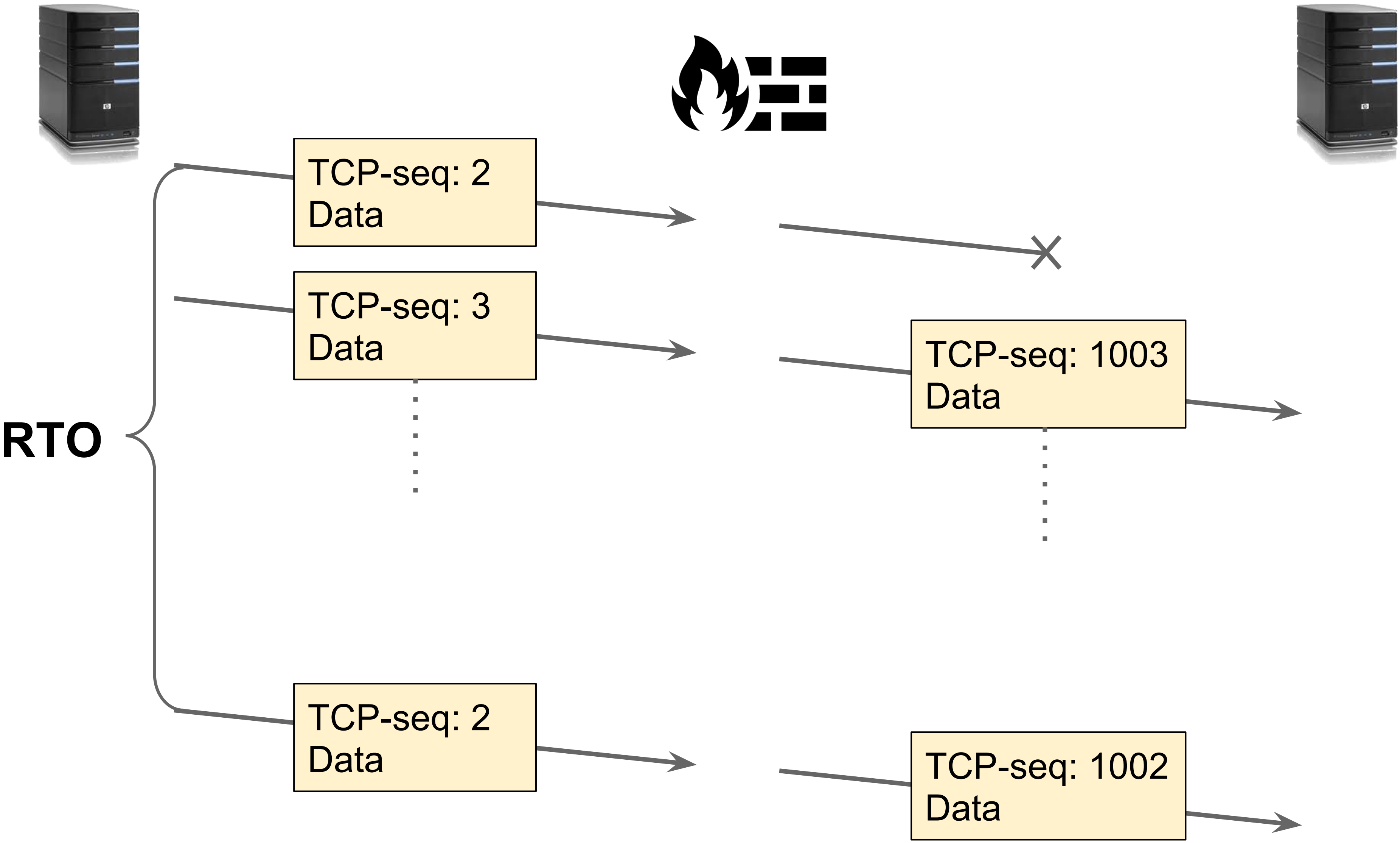
TCP-ack: 2

TCP-ack: 1002
SACK: 1003 - 1010

TCP-ack: 2
SACK: 1003 - 1010

Discard SACK-info
Disregard duplicate ACK

Sequence number randomization



RTO

TCP-seq: 2
Data

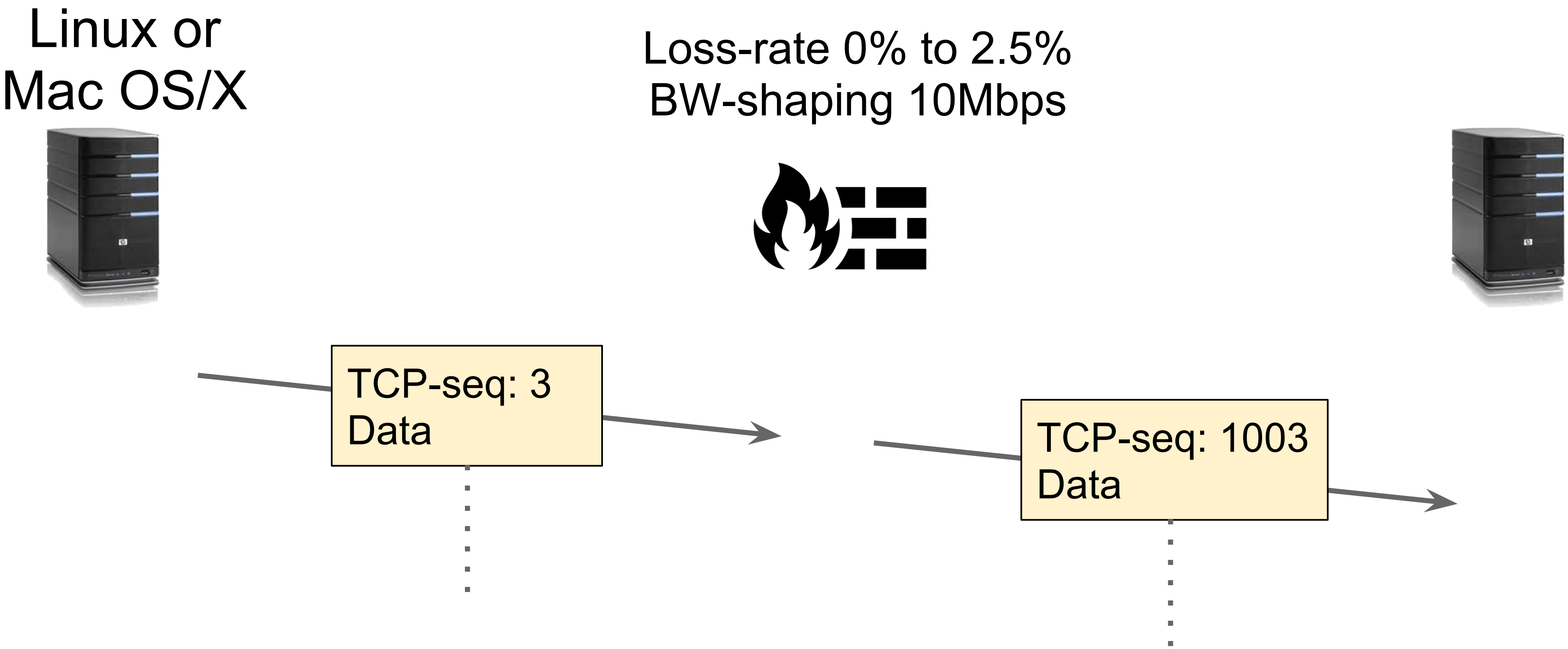
TCP-seq: 3
Data

TCP-seq: 1003
Data

TCP-seq: 2
Data

TCP-seq: 1002
Data

Performance Impact



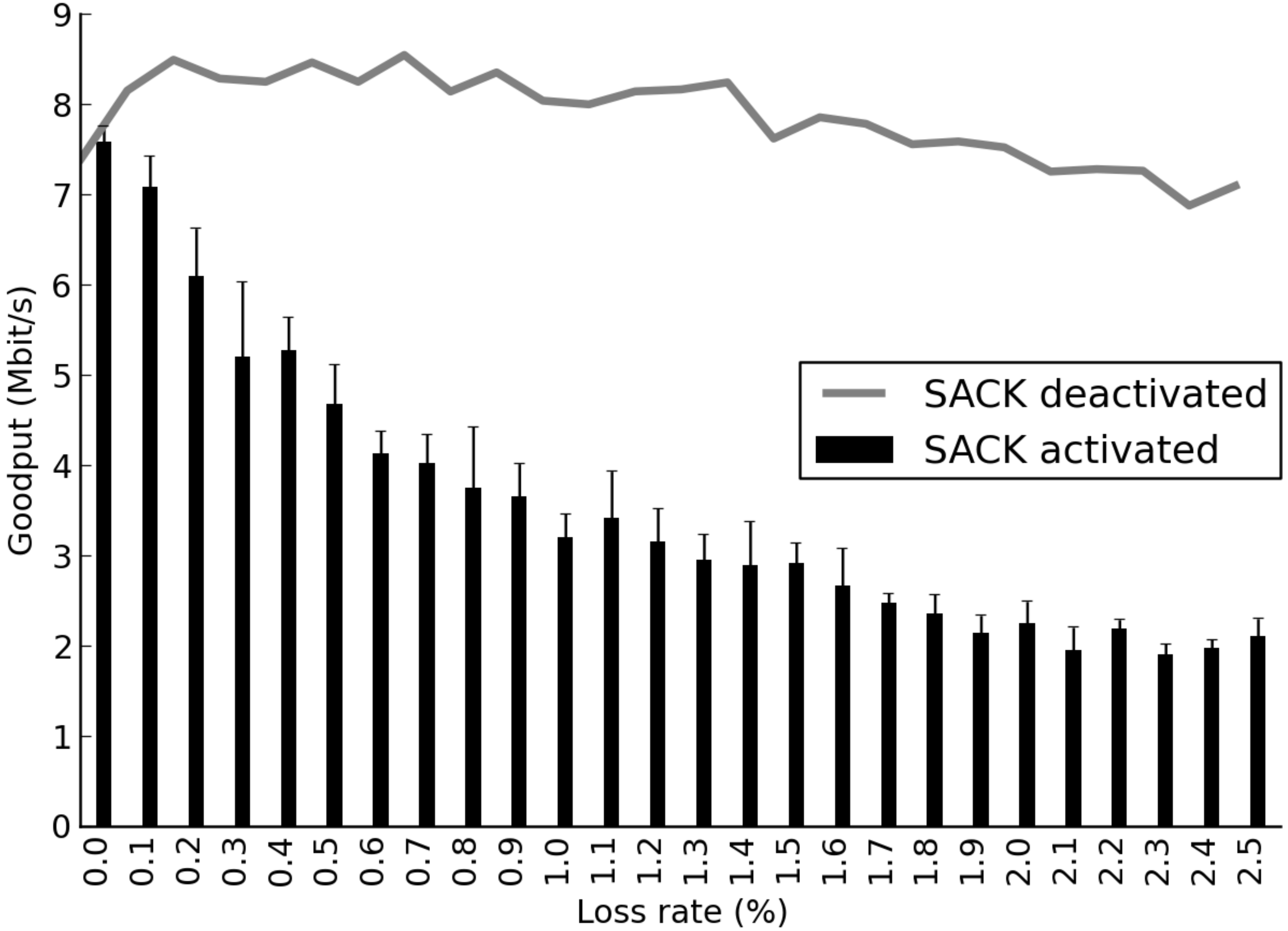
Linux or Mac OS/X

Loss-rate 0% to 2.5%
BW-shaping 10Mbps

TCP-seq: 3
Data

TCP-seq: 1003
Data

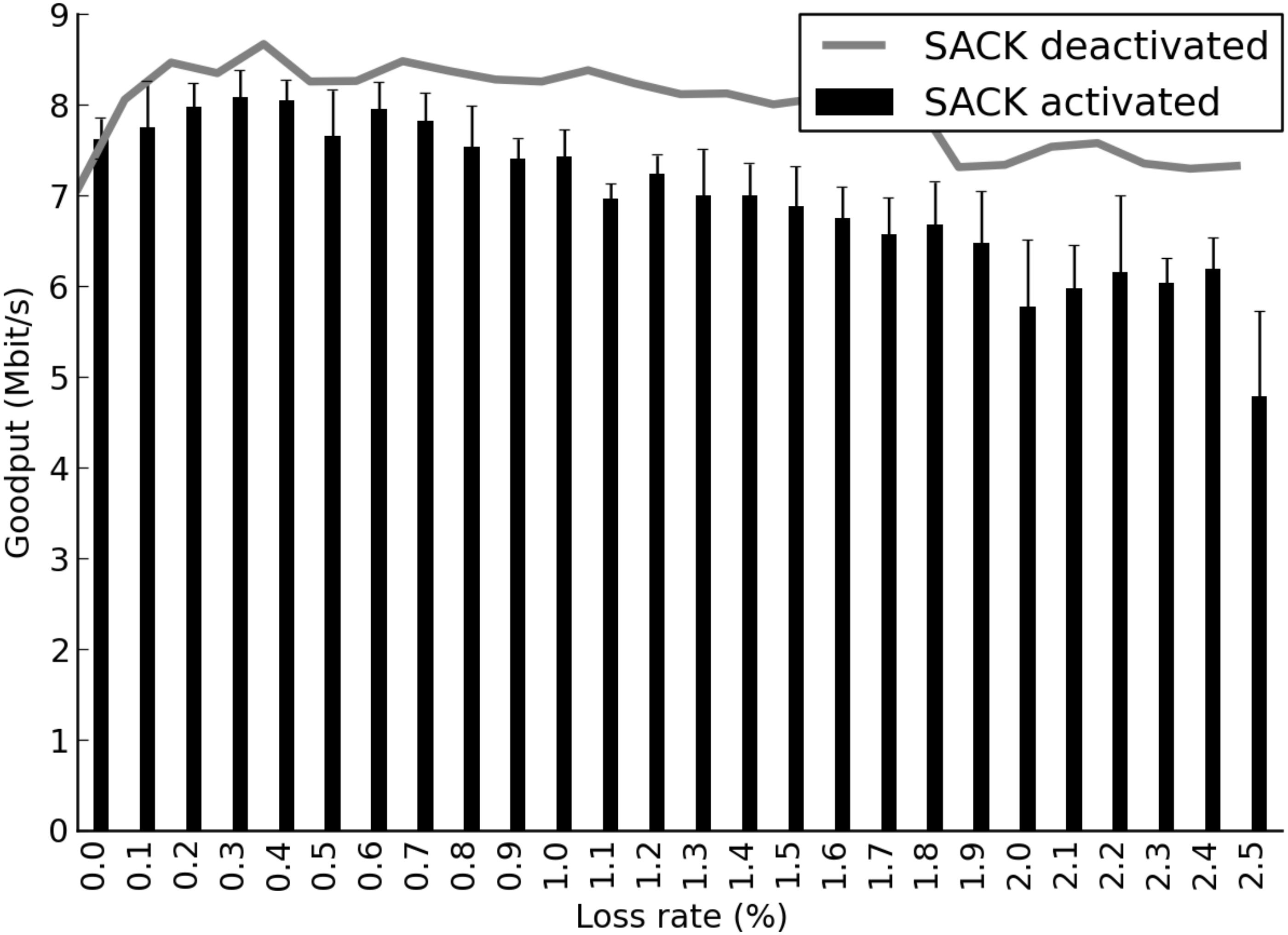
Performance Impact



Accounting duplicate ACKs

Duplicate ACKs with invalid SACK-blocks should not be discarded.

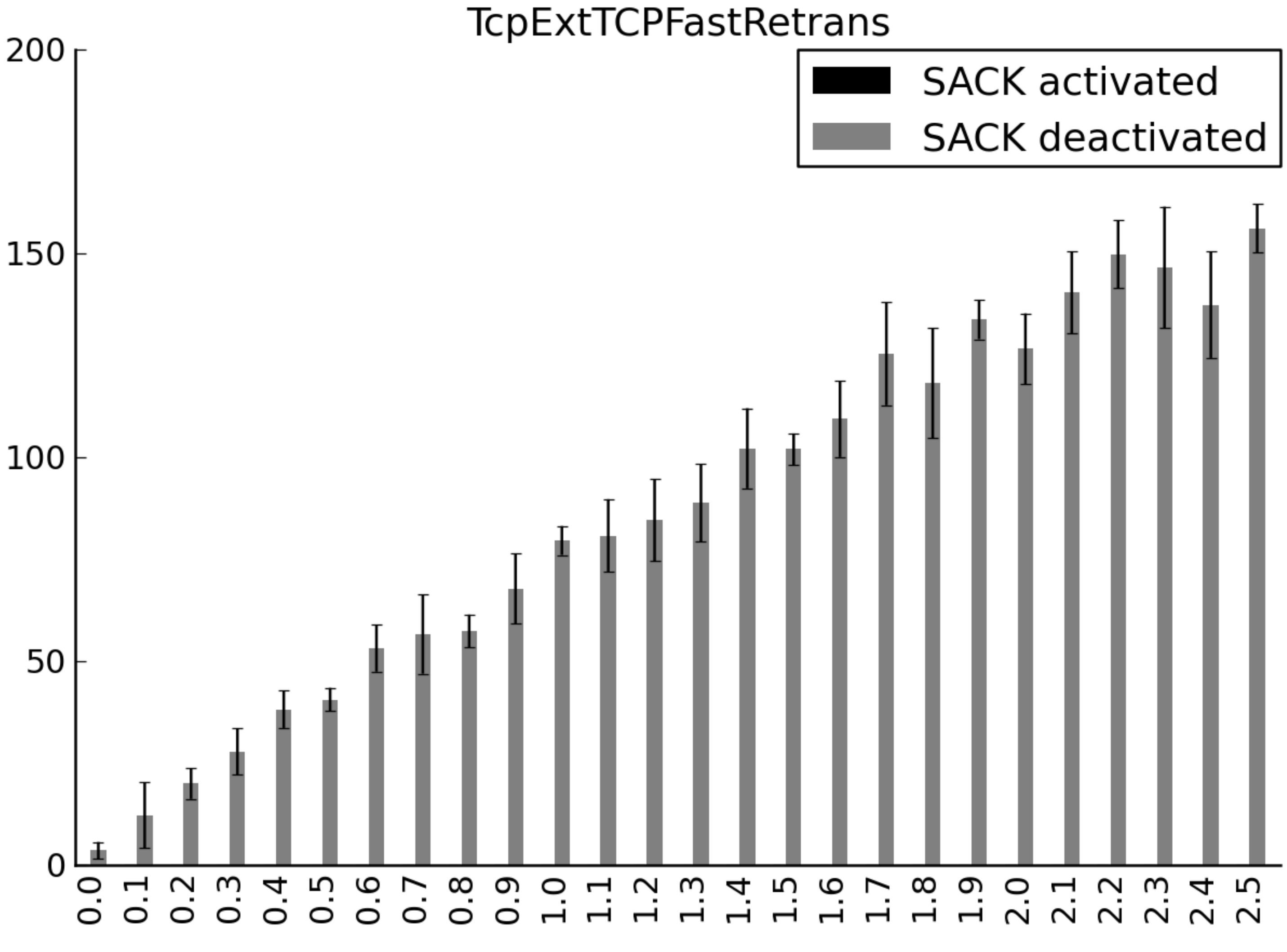
Accounting duplicate ACKs



Conclusion

- Middleboxes mess up with our connections
- And they probably won't go away
- We have to consider the middleboxes in our protocols

Fast-Retransmission counters


TcpExtTCPFastRetrans
SACK activated
SACK deactivated