# Non-Renegable Selective Acknowledgments for TCP

Fan Yang, Paul D. Amer
CIS, University of Delaware, USA

Si Quoc Viet Trang, **Emmanuel Lochin**
ISAE, Université de Toulouse

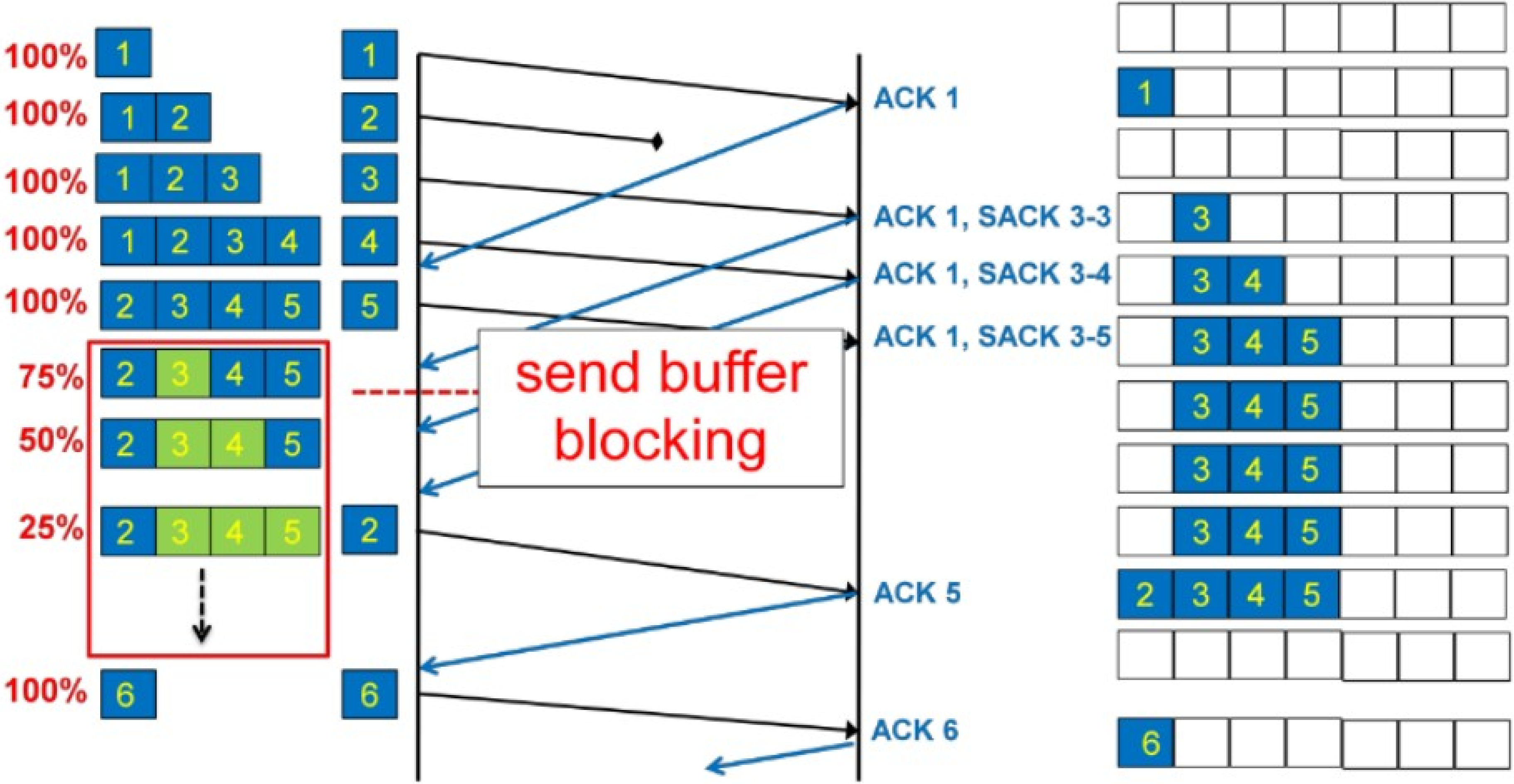
# Background

- TCP is designed to tolerate reneging
  - The possibility of reneging forces a transport sender to maintain copies of SACKed data in the send buffer until they are cumulatively acked
- This design has been challenged since :
  - reneging rarely occurs in practice
    - *N. Ekiz, Transport Layer Reneging, PhD Univ. Delaware, 2012*
  - even when reneging does occur, it alone generally does not help the operating system resume normal operation when the system is starving for memory

# TCP/SACK

- TCP send buffer gives a window of contiguous bytes to transmit
- The lower edge of the window is defined by the received highest cumack number
- The upper edge is defined to be the highest cumack number plus the number of bytes in the advertised receive window
- Under these two circumstances, there is no advantage to having a receive window larger than the send window

# Normal TCP data transfer

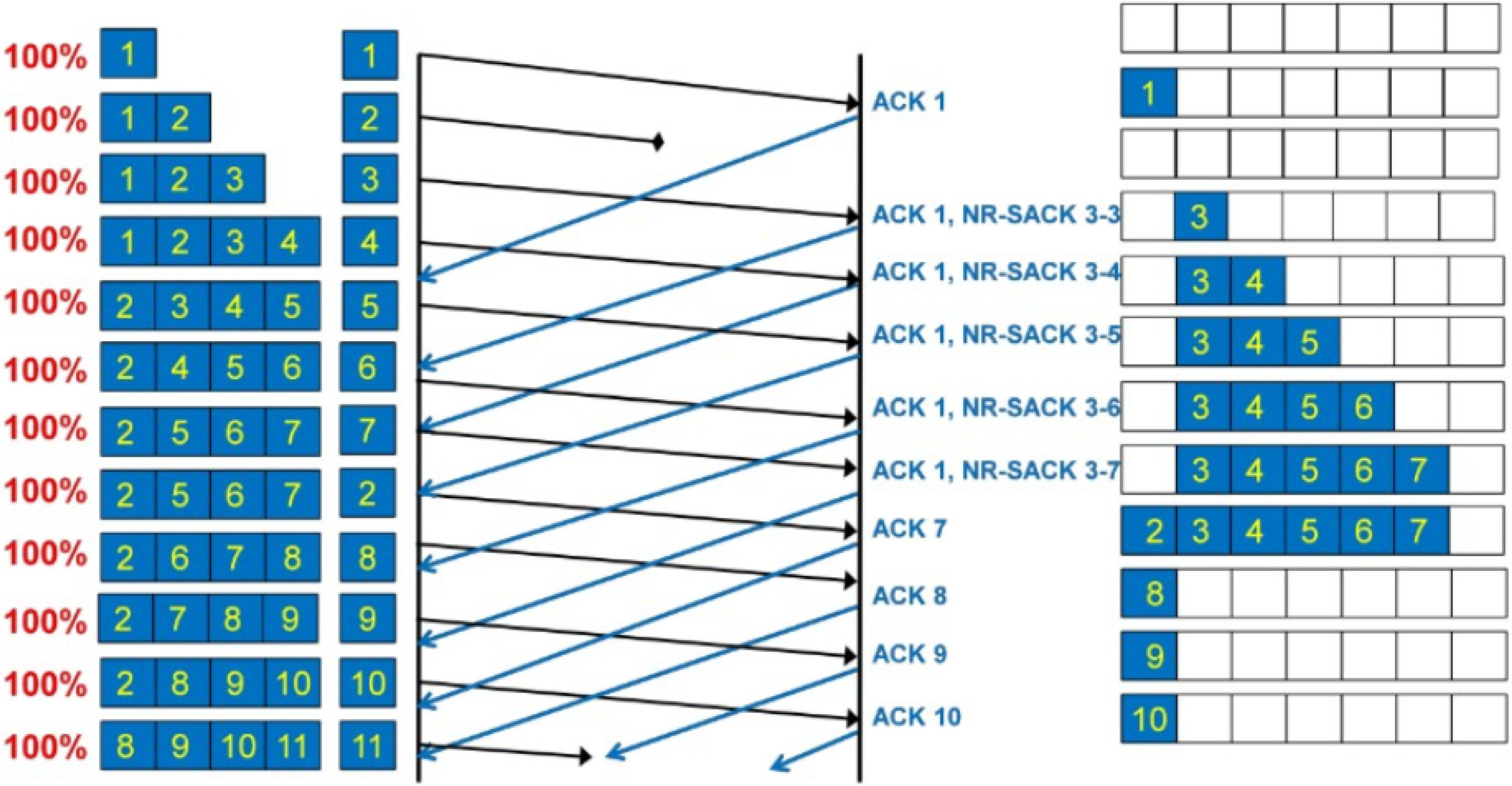


Sender buffer == 4

Receiver buffer == 7

# TCP NR-SACK

## (Non-Renegable Selective Acknowledgments)



# NR-SACK implementations

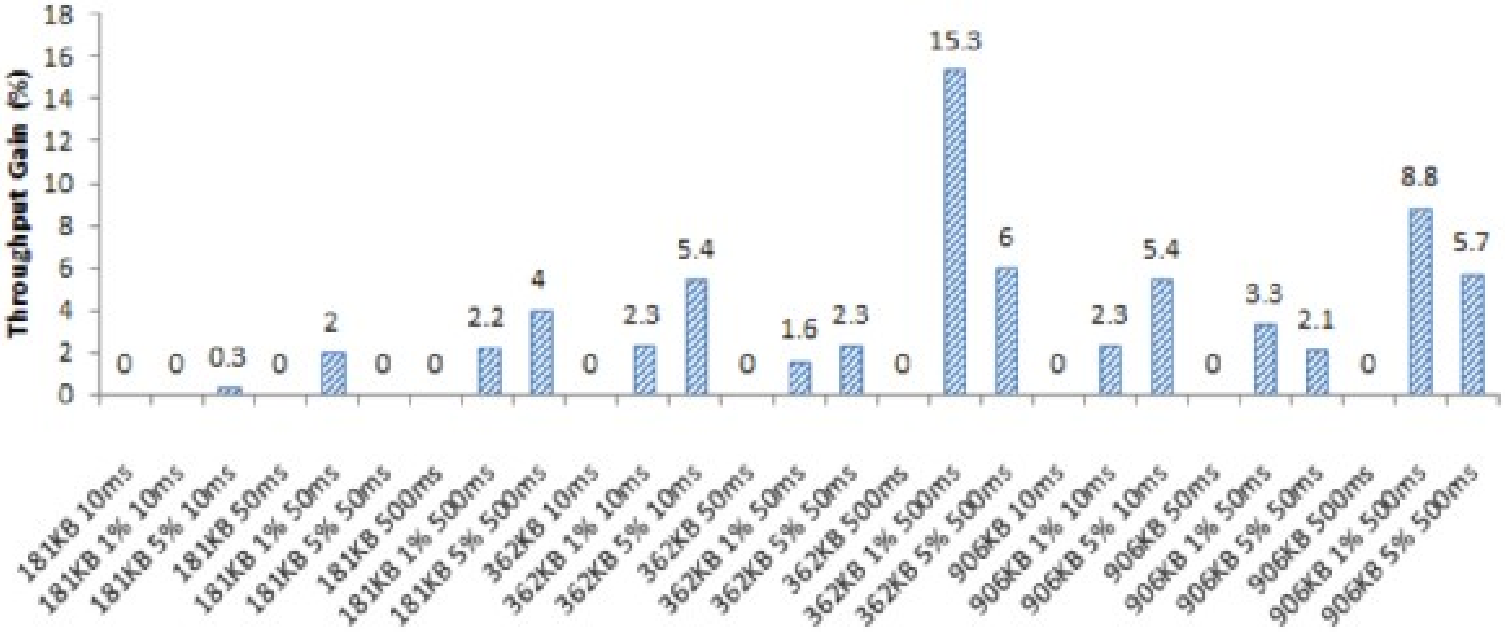
- Severals studies show that NR-SACK for SCTP and MPTCP not only reduce sender's memory requirements but also improve the end-to-end throughput under certain conditions
- QUIC implements an NR-SACK-like mechanism
- So … does TCP still need NR-SACK ?
- Does freeing received out-of-order PDUs from the send buffer by using NR-SACKs can improve end-to-end performance ?

# … and the answers are

- This improvement results when send buffer blocking occurs in TCP
- TCP data transfers with NR- SACK never perform worse than those without NR-SACK
- NR-SACKs can improve end-to-end throughput when send buffer blocking occurs
- Under certain circumstances, we observe throughput increasing by using TCP NR-SACK as much as 15%

# Some results

- NR-SACK implemented in GNU/Linux kernel
  - Quite tricky to allow a possibly non-contiguous set of bytes
- In GNU/Linux the default upper limit of the TCP send buffer size is 905KB
- A smaller send buffer, which needs not to keep copies of SACKed data, can keep a larger receive window busy (e.g., default send and receive buffer sizes for Linux 2.6.31 are 16,384 and 87,380 bytes, respectively)



# Thanks for your attention

- Implementation of NR-SACK for TCP realized by Fan Yang from University of Delaware

yangfan@udel.edu