NR-SACKs for SCTP (Non-Renegable SACKs)

Preethi Natarajan
Nasif Ekiz
Ertugrul Yilmaz
Paul Amer

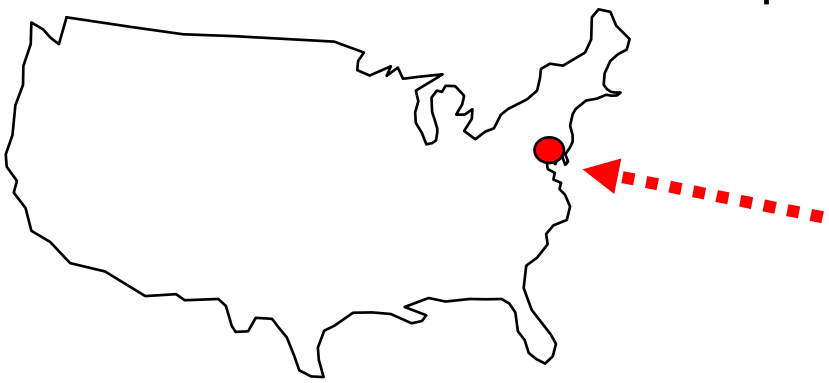
University of Delaware

Randall Stewart

The Research Group

Janardhan Iyengar

Franklin & Marshall College

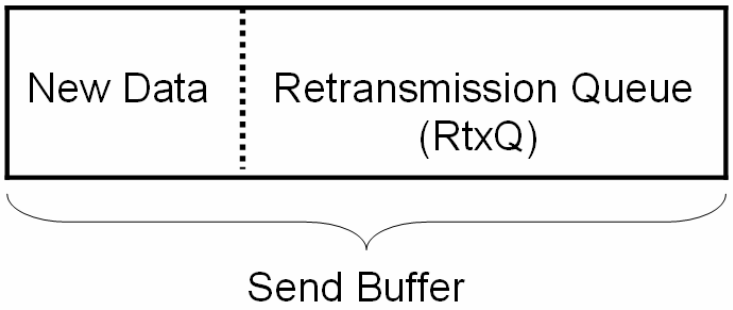


Dela-where?

Agenda

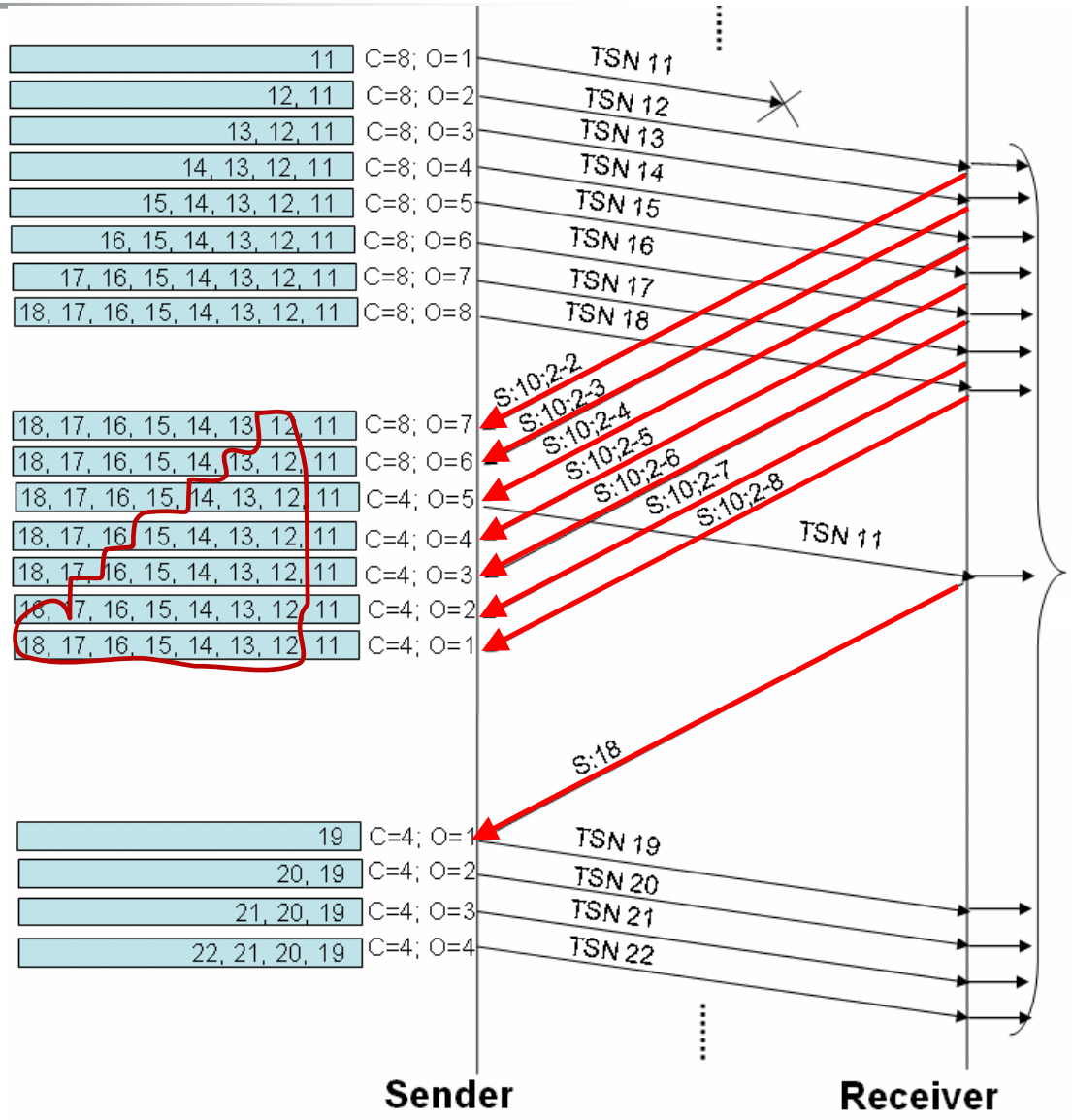
- Reneging and SCTP SACKs
- Example
- Details of Non-Renegable SACKs
  - SCTP Results
- Introduce Concurrent Multipath Transfer (CMT)
  - Results

Reneging and SCTP SACKs



- Receiver cannot renege on cum-acked / delivered data
  - sender discards cum-acked data from rtxq
- Receiver may renege on gap-acked (out-of-order) data
  - due to buffer overbooking
  - sender does not discard gap-acked data

Unordered Data Transfer using SACKs

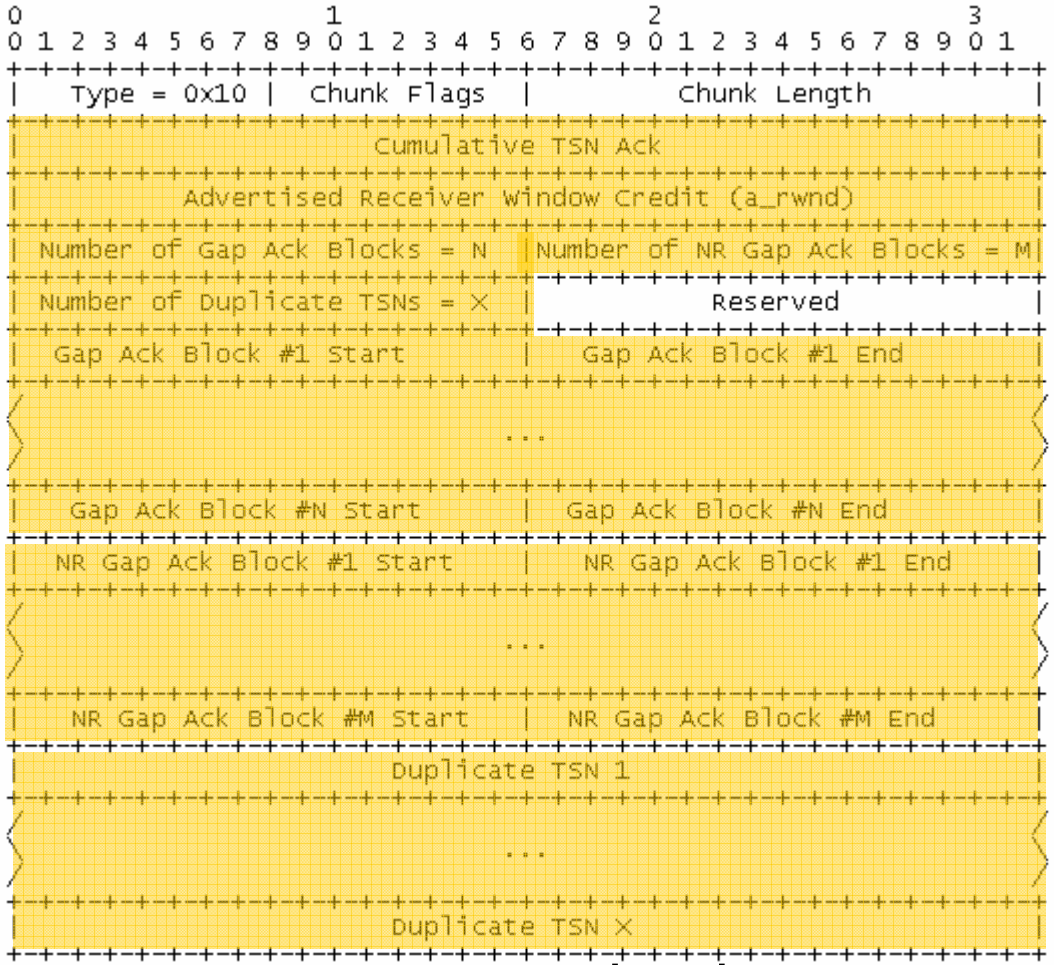


"Unnecessary"; delivered to app, but still in RtxQ

Delivered to App

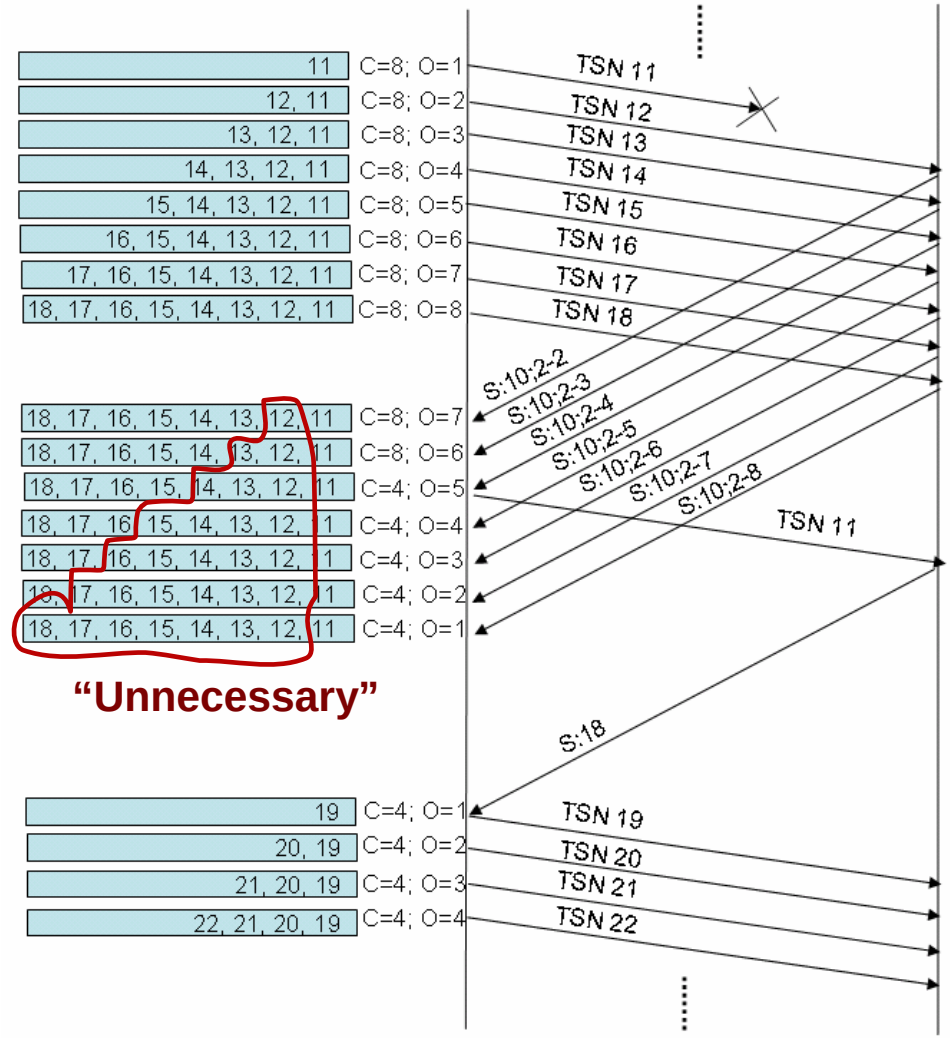
IETF ID: NR-SACKs for SCTP

- Peers negotiate NR-SACK capability during association establishment
- Replace SACKs with NR-SACKs during transfer
- Congestion and flow control similar to SACKs
- NR-SACKs contain non-renegable gap-ack blocks; sender can use this info to free buffer



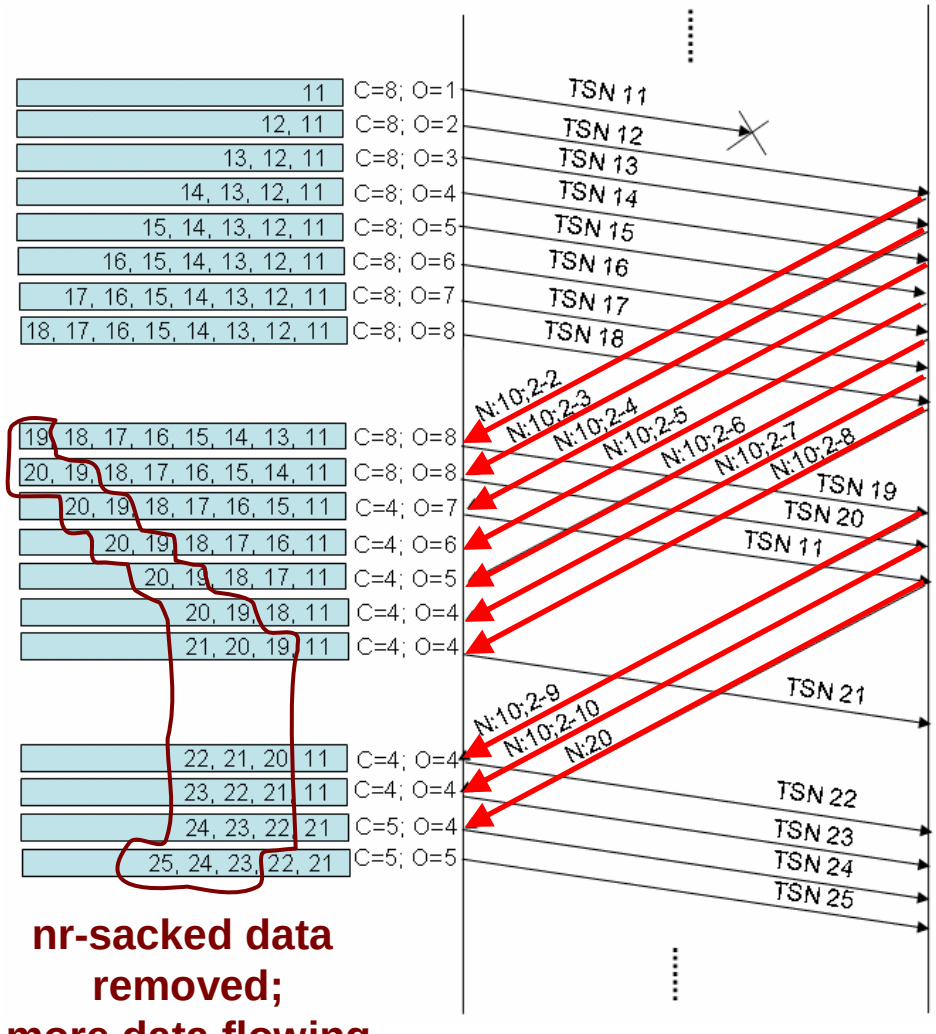
NR-SACK Chunk

Unordered Data Transfer



"Unnecessary"

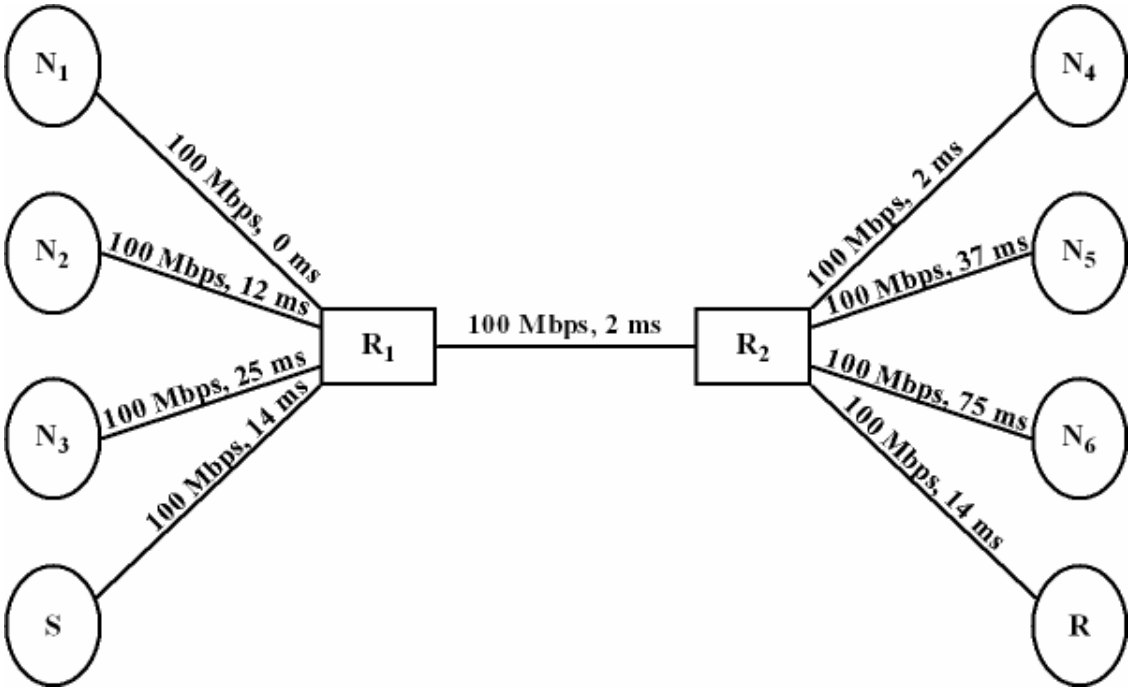
With SACKs



nr-sacked data removed; more data flowing

With NR-SACKs

Simulation Setup



- L. Andrew et. al., TCP Evaluation Suite, in PFLDNet 2008
  - Dumb-bell access link topology
  - Application-level cross-traffic generation
- Unordered SCTP transfer using SACKs vs. NR-SACKs

Efficient RtxQ Utilization

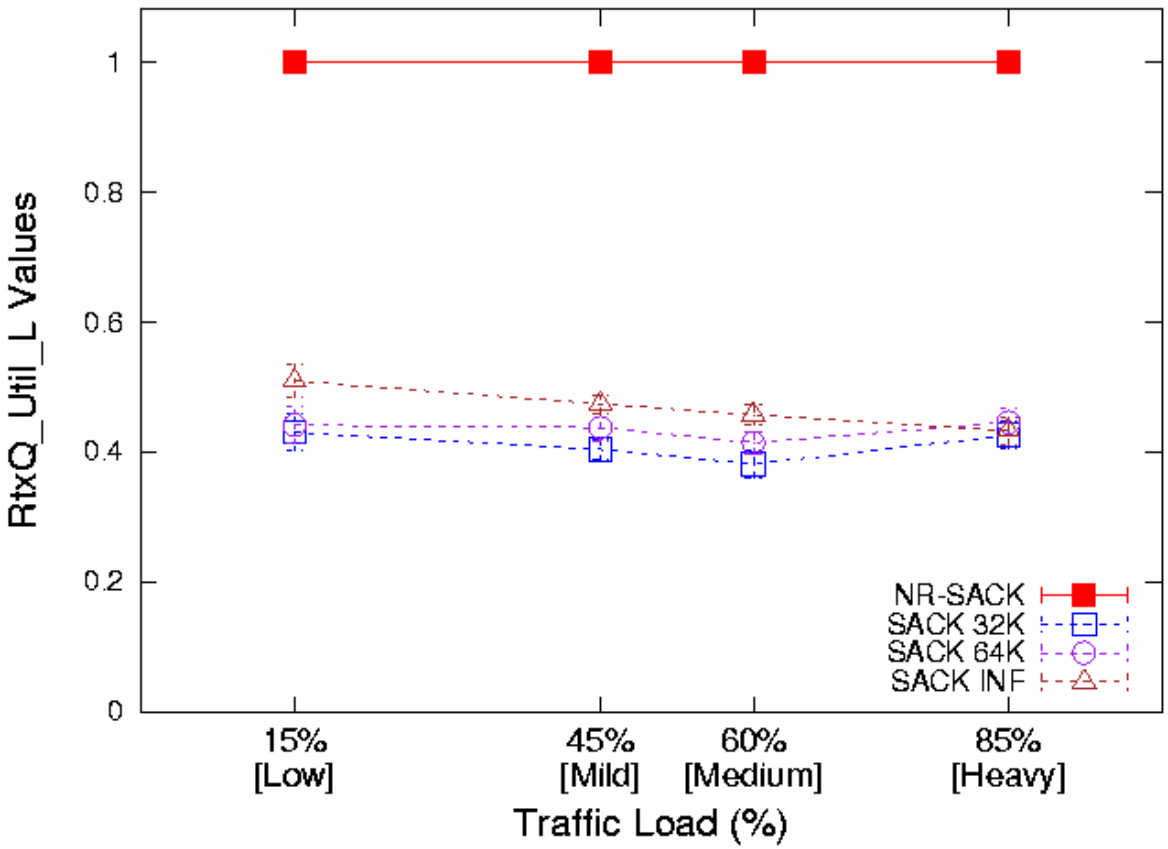
- Two types of data in RtxQ
  - “Unnecessary”: Data received and non-renegable
  - “Necessary”: (i) data in-flight, (ii) data received and renegable
  - RtxQ is most efficiently utilized when all data in rtxq are “necessary”

- Efficient RtxQ Utilization (RtxQ_Util)

$$\text{RtxQ\_Util} = \frac{\text{“Necessary”}}{\text{“Necessary”} + \text{“Unnecessary”}}$$

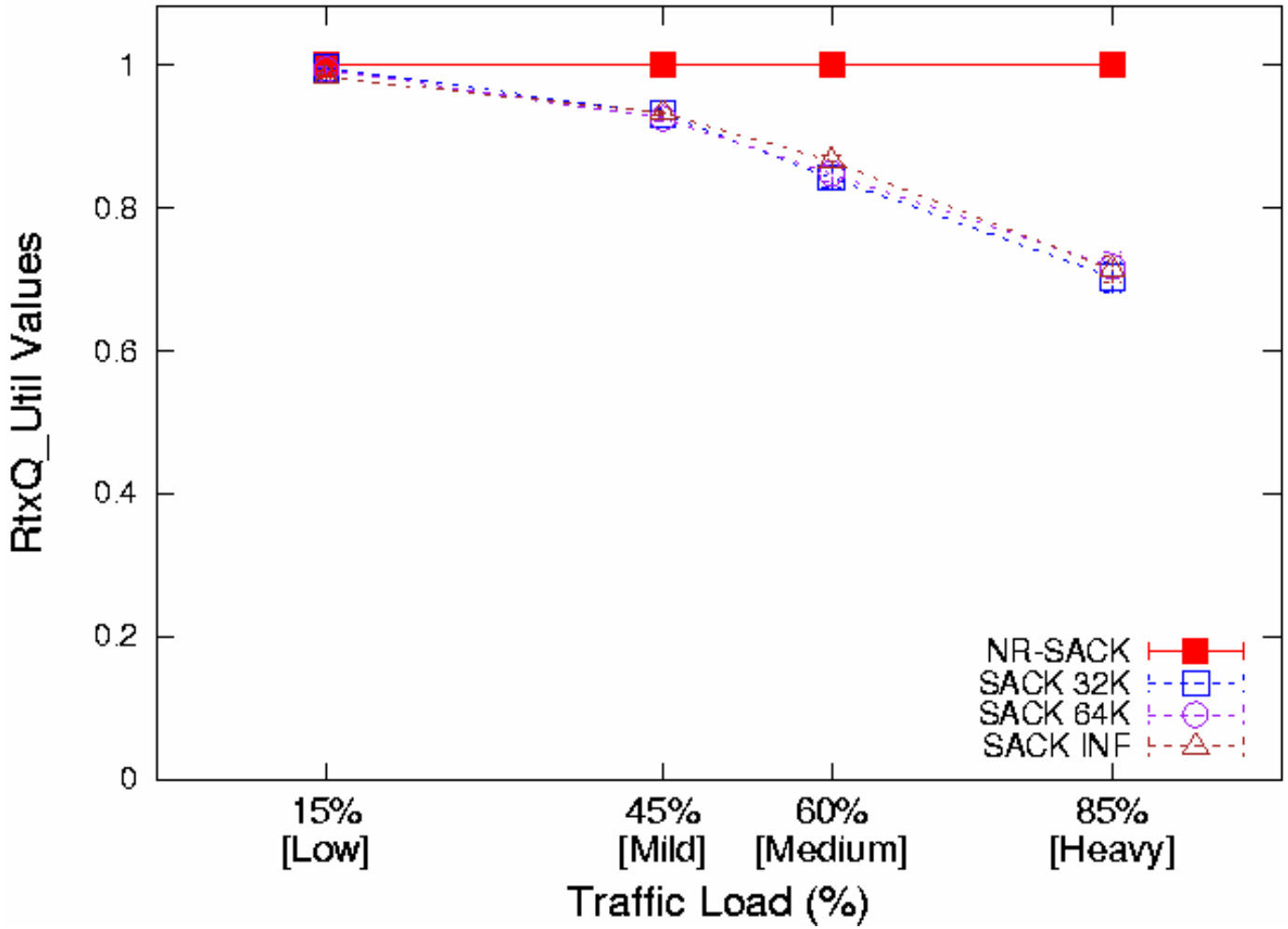
RtxQ_Util_L: Utilization during Loss Recovery

SACKs and NR-SACKs differ in RtxQ_Util only when data received out-of-order — during loss

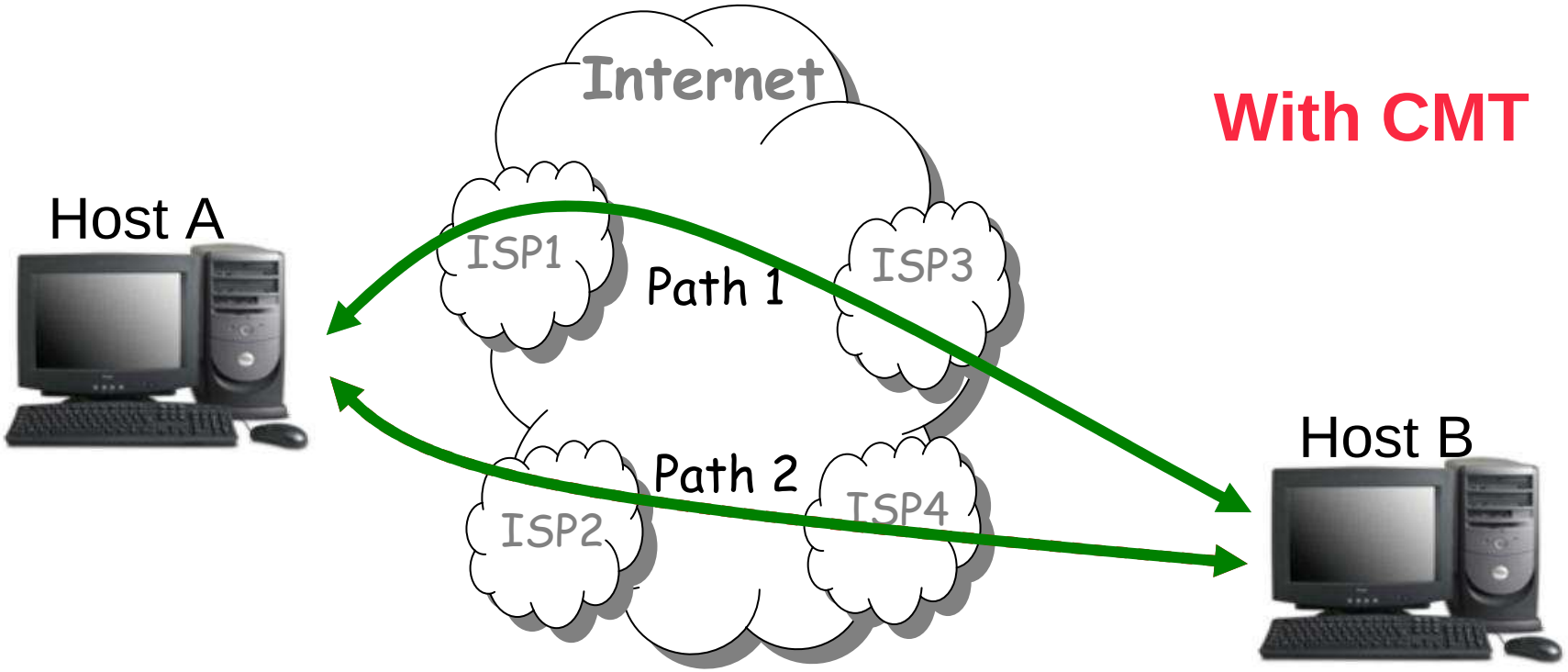


- SCTP with SACKs wastes ~50% of rtxq during loss recovery
- NR-SACKs always utilize rtxq most efficiently

RtxQ_Util for SCTP

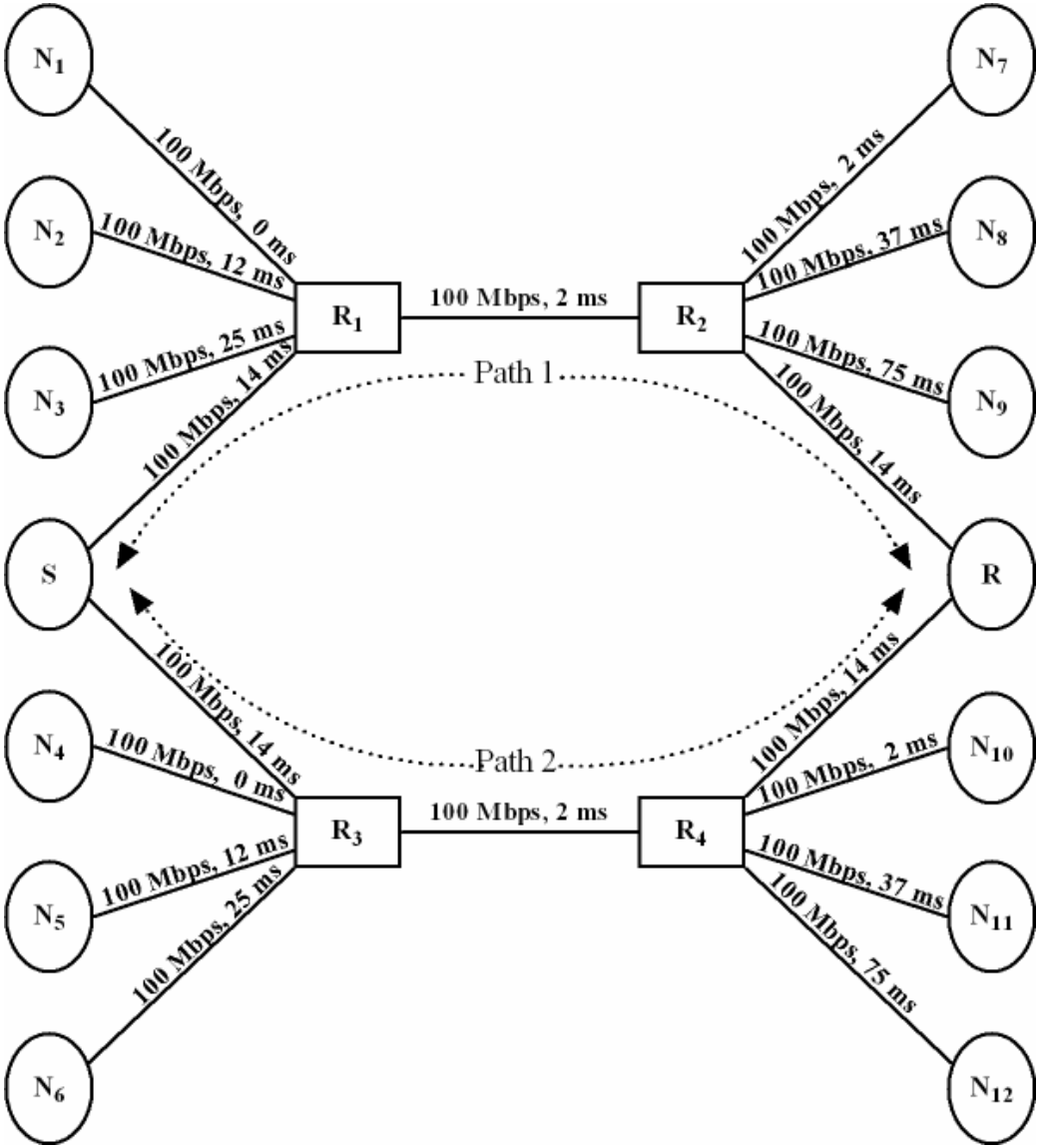


What is Concurrent Multipath Transfer (CMT)?

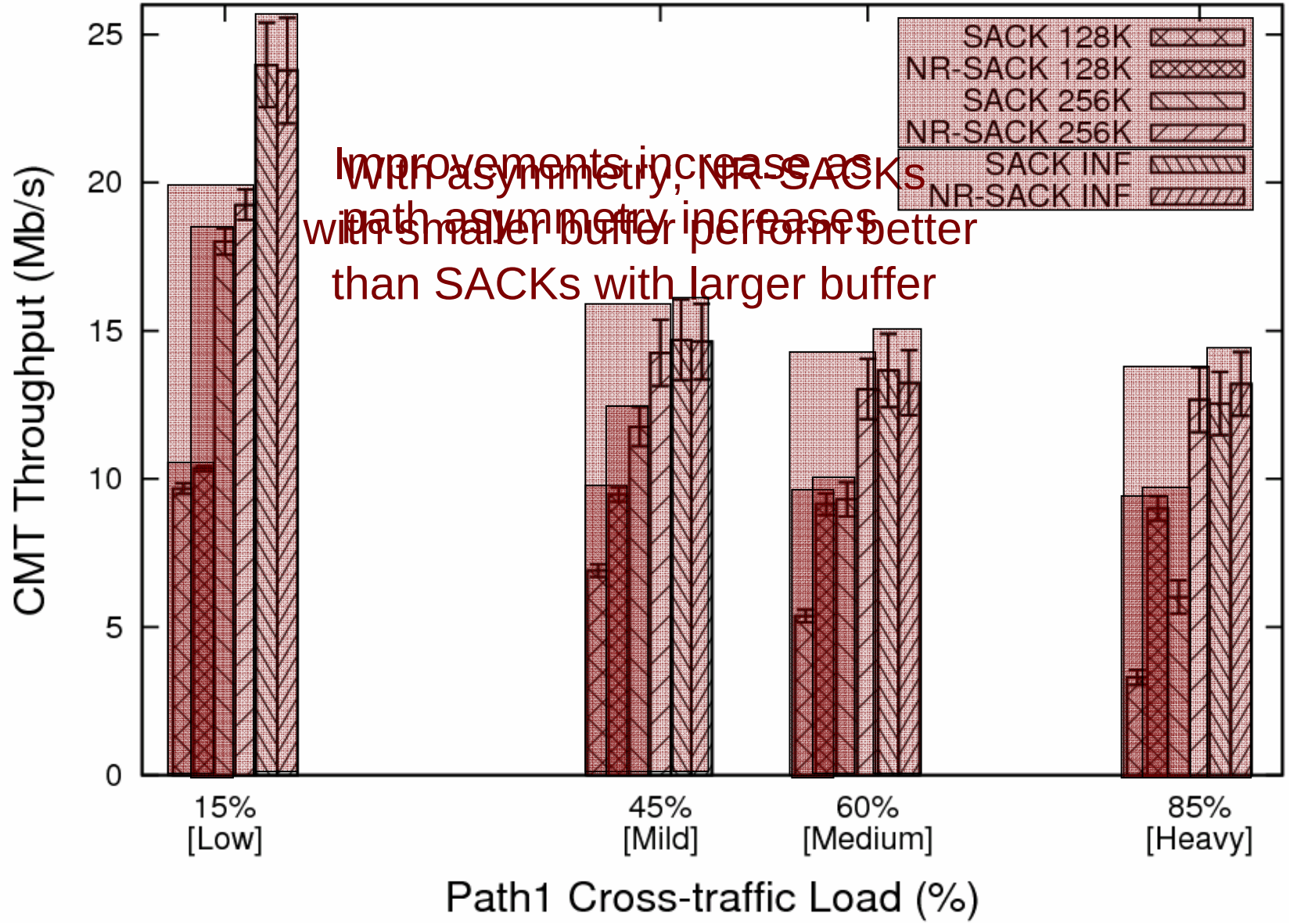


Asymmetric path characteristics increase data reordering in CMT

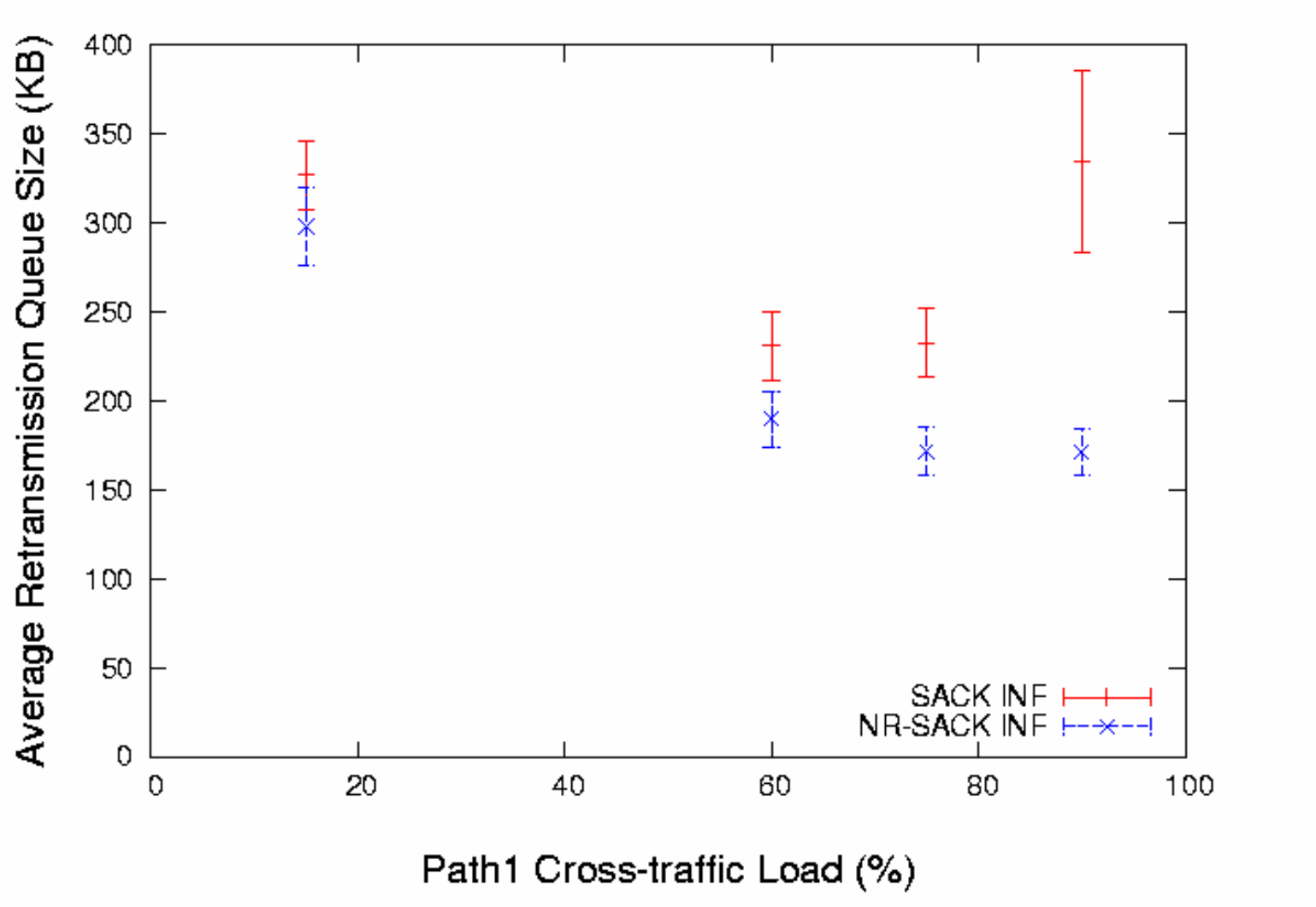
Simulation Setup for CMT



NR-SACKs Reduce Send Buffer Blocking and Improve CMT Throughput



But, NR-SACKs Require Smaller RtxQ



Summary

- SACKs vs. NR-SACKs
  - SACKs waste RtxQ (kernel memory) when out-of-order data is non-renegable
  - NR-SACKs utilize RtxQ more efficiently
  - NR-SACKs reduce send buffer blocking and improve CMT throughput
- Simulation results presented at IETF Dublin
- Currently implementing NR-SACKs in FreeBSD

Thank You

- Questions…

Retransmission Queue (Rtxq) Utilization

- Efficient utilization at time $t = k/r$
  - k = # necessary (in flight or renegable) TPDUs in the rtxq
  - r = size of the rtxq

- Efficient utilization for the entire file transfer

$$RtxQ\_Util = \left(\sum t_i \times \frac{k_i}{r_i} \right) \div T; \left(\sum t_i = T \right)$$

  - k_i/r_i = efficient rtxq utilization during t_i
  - T = File Transfer time