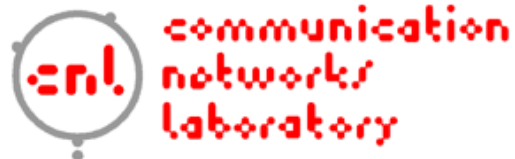




TCP with adaptive delay and loss response for heterogeneous networks

Modupe Omueti and Ljiljana Trajković
{momueti, ljilja}@cs.sfu.ca

Communication Networks Laboratory
<http://www.ensc.sfu.ca/research/cnl>
School of Engineering Science
Simon Fraser University



A decorative graphic on the left side of the slide, featuring overlapping yellow, red, and blue squares with a black crosshair.

Roadmap

- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR) algorithm
- Performance evaluation of TCP-ADaLR:
 - simulation scenarios and results
 - fairness and friendliness scenarios
- Conclusions



Introduction

- Transmission control protocol (TCP):
 - provides byte-stream transport for most Internet applications such as remote login, FTP, and HTTP
 - carries up to 90% of Internet traffic
 - originally designed for wired networks characterized by negligible bit error rates
- The Internet:
 - growth in wireless IP communications
 - increasing demand in multimedia and data applications

M. Fomenkov, K. Keys, D. Moore, and K. Claffy, "Longitudinal study of Internet traffic in 1998-2003," in *Proc. ACM Winter Int. Symp. Inf. and Commun. Technologies*, Cancun, Mexico, Jan. 2004, pp. 1–6.

IP: Internet Protocol

FTP: file transfer protocol

HTTP: hyper-text transfer protocol



Heterogeneous networks with GEO satellite



- Provide global Internet services for areas with limited or no terrestrial cable infrastructure
- Offer **high data rates** of the order of **1 Mb/s or higher** through high-bandwidth **geostationary earth orbit (GEO)** satellite links
- Possess scalable architecture, multicast capabilities, and large coverage areas
- Incur lower risks in development compared to satellite constellations of non-GEO satellites

A. Jamalipour, M. Marchese, H. Cruickshank, J. Neal, and S. Verma, "Broadband IP networks via satellites-part II," *IEEE J. Select. Areas Commun.*, vol. 22, no. 3, pp. 433–437, Apr. 2004.

R. A. Peters and M. Farrell, "Comparison of LEO and GEO satellite systems to provide broadband services," in *Proc. 21st AIAA Int. Commun. Satellite Syst. Conf. and Exhibit*, Yokohama, Japan, Apr. 2003, AIAA–2003–2246.

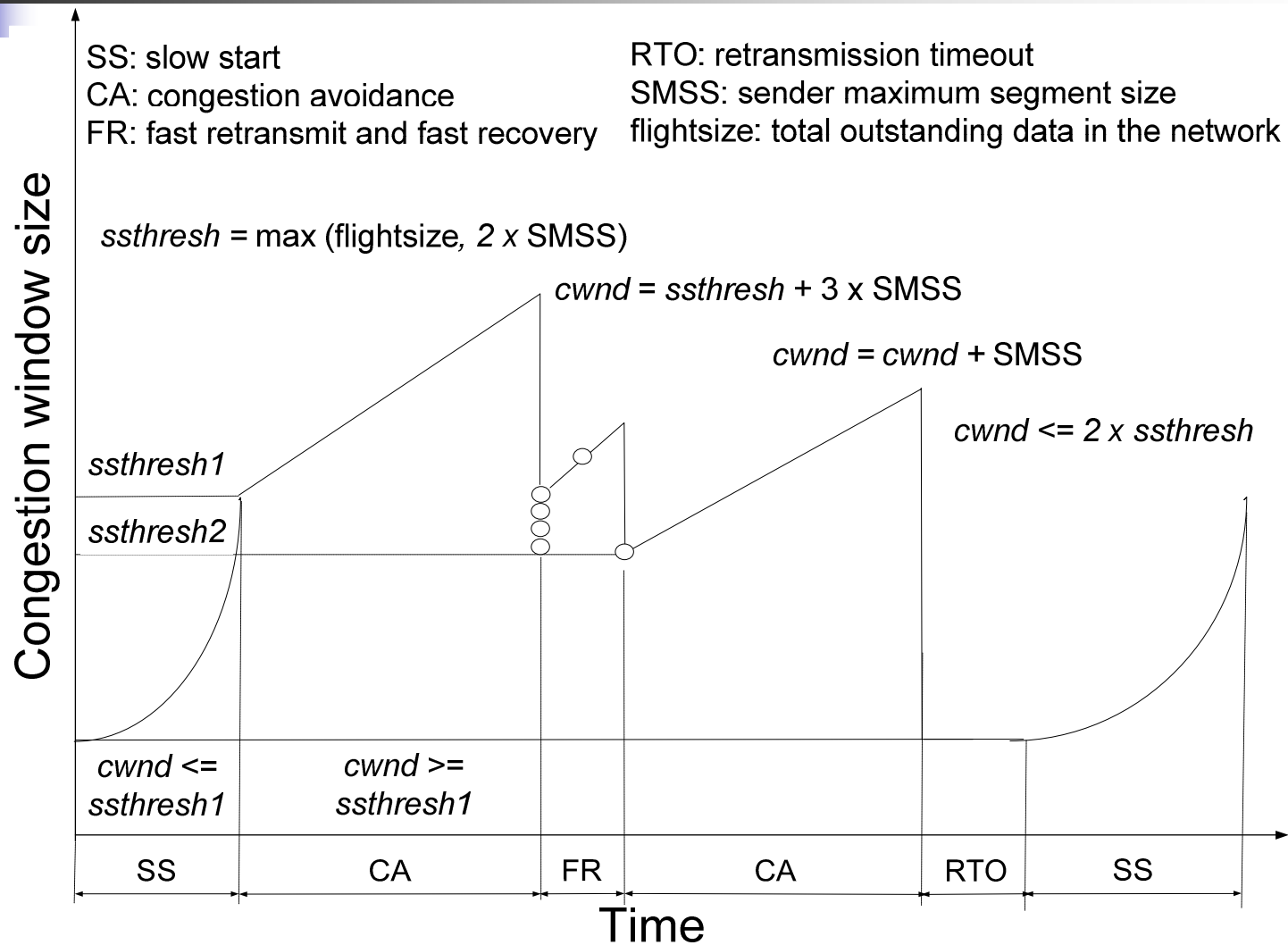
A decorative graphic on the left side of the slide, featuring overlapping yellow, red, and blue squares with a black crosshair.

Roadmap

- Introduction
- **Background and related work**
- TCP with adaptive delay and loss response (TCP-ADaLR) algorithm
- Performance evaluation of TCP-ADaLR:
 - simulation scenarios and results
 - fairness and friendliness scenarios
- Conclusions



TCP congestion control algorithms





TCP delayed acknowledgement option

- Allows TCP receivers to send an acknowledgement (ACK) for every second consecutive full-sized segment received
- Implemented by many TCP receivers in the Internet:
 - default interval period: 200 ms
 - maximum interval period: 500 ms
- Reduces protocol processing overhead
- Increases network efficiency and maximizes network bandwidth

J. Chen, Y. Z. Lee, M. Gerla, and M. Y. Sandidi, "TCP with delayed ACK for wireless networks," in *Proc. IEEE/CreateNet BROADNETS 2006*, San Jose, CA, USA, Oct. 2006, pp. 1–6.

W. Lilakiatsakun and A. Seneviratne, "TCP performances over wireless links deploying delayed ACK," in *Proc. 57th IEEE Veh. Technol. Conf.*, Jeju, Korea, Apr. 2003, vol. 3, pp. 1715–1719.

A full-sized segment is equivalent to the sender maximum segment size (SMSS)



GEO satellite links: characteristics

- Long propagation delay:
 - ~ 250 ms one-way due to high GEO satellite altitude
 - ≥ 500 ms RTT depending on satellite inclination
- Large bandwidth delay product BDP:
 - BDP defines the amount of data a protocol should have unacknowledged in order to fully utilize link capacity
 - computed as product of satellite link capacity and RTT
 - long propagation delay results in large BDP
- High bit error rate $\sim 10^{-6}$ without forward error correction
- Asymmetry path (different downlink and uplink channel capacities)

RTT: round trip time

Impact of GEO satellite link characteristics on TCP performance



- **Long propagation delays** prevent TCP from achieving maximum throughput during slow start phase
- **Large BDP** values require large number of unacknowledged bytes in flight
- **High BERs** cause packet losses:
 - packet losses misinterpreted as congestion indication
 - *cwnd* reduced leading to TCP performance degradation
- **Path asymmetry** results in traffic burstiness:
 - ACK losses may occur on low-bandwidth downlink paths
 - ACK losses misinterpreted as congestion indication

BDP: bandwidth delay product
BER: bit error rate
ACK: acknowledgement



Related work

- **End-to-end** solutions preserve the end-to-end semantics of TCP: TCP SACK, TCP-Peach, TCP Westwood, TCP-Star, TCP-Hybla, and TCP NewVegas
- **Split connection** solutions shield satellite link characteristics from the terrestrial segment:
 - split TCP connections at an intermediate node
 - violate end-to-end semantics of TCP
- **Link layer** solutions:
 - TCP-aware modify TCP header information
 - TCP-unaware employ FEC and ARQ techniques for detecting and retransmitting lost segments at the link layer



Roadmap

- Introduction
- Background and related work
- **TCP with adaptive delay and loss response (TCP-ADaLR) algorithm**
- Performance evaluation of TCP-ADaLR:
 - simulation scenarios and results
 - fairness and friendliness scenarios
- Conclusions

TCP-ADaLR: TCP with adaptive delay and loss response



- End-to-end solution for improving TCP performance in broadband GEO satellite networks:
 - scaling component ρ
 - adaptive *cwnd* increase mechanism
 - adaptive *rwnd* increase mechanism
 - loss recovery mechanism
- Requires modifications only at the TCP sender
- Considers the case with the delayed ACK option enabled
- Implemented in OPNET modeler v. 11.0.A
 - extension to TCP SACK
 - applicable to TCP NewReno

TCP-ADaLR: TCP with adaptive delay and loss response



- Designed for satellite networks that exhibit:
 - high BER losses
 - long propagation delays
- Improves TCP performance by considering the effect of RTT on the *cwnd* increments:
 - computes **scaling component ρ** from measured RTT
 - adaptively changes *cwnd* based on the value of **scaling component ρ**
- Allows faster transmission of additional segments both in the presence and absence of losses

BER: bit error rate
RTT: round trip time
cwnd: congestion window



Scaling component ρ

- Used to increase the *cwnd* increment during the slow start and congestion avoidance phases
- Calculated as:
 - $\rho = (\textit{sampleRTT s}/1 \textit{ s}) \times 60$
 - *sampleRTT* is normalized by 1 s
 - fixed parameter 60 is the minimum recommended value for the maximum RTO *rto_max*
 - lower bound: 1
 - upper bound: 60
- Mitigates the negative effect of the long propagation delay on achieving high transmission rates rapidly

sampleRTT: the measured RTT of a data segment sample not retransmitted

rto_max: the upper limit on the interval that a TCP sender waits before retransmission

Adaptive *cwnd* increase mechanism: slow start phase



- Based on ρ and the presence or absence of losses
- Slow start phase is divided into **four sub-phases** depending on current *cwnd* and the *flightsize*:
 - **four slow start sub-phases** are selected based on the ratio of the initial value of *ssthresh* 64 KB and largest initial *cwnd* 16 KB
 - during four **slow start sub-phases** increment *cwnd* by:
 - $(\sqrt{\rho} / 4) \times \text{SMSS}$, if no losses occur and $\rho \geq 15$
 - **SMSS**, if losses occur as in conventional TCP
 - at all other times, increment *cwnd* by **SMSS**

flightsize: total outstanding unacknowledged data in the network
SMSS: sender maximum segment size

Adaptive *cwnd* increase mechanism: heuristics



- $\rho \geq 15$ corresponds to an $RTT \geq 250$ ms
- selected based on simulation of an FTP file download for various RTTs
- $(\sqrt{\rho} / 4)$:
 - is equivalent to a value between $(1 - 2) \times SMSS$
 - prevents large line-rate bursts

Download response time for a 50 MB file

RTT (ms)	FTP download response time (s)
25	251.8
50	252.1
100	252.5
200	253.5
250	272.7
500	470.1

M. Allman, "TCP congestion control with appropriate byte counting (ABC)," *IETF RFC 3465*, Feb. 2003.

Adaptive *cwnd* increase mechanism: congestion avoidance phase



- increment *cwnd*:
 - $(\sqrt{\rho} / 2) \times \text{SMSS} \times \text{SMSS} / \text{cwnd}$:
 - if losses occur, TCP sender is out of fast recovery, and $\rho \geq 15$
 - if *flightsize* is less than $\text{rwnd} / 2$ and $\rho \geq 15$
 - $\text{SMSS} \times \text{SMSS} / \text{cwnd}$ (linearly), as in conventional TCP at all other times
- $(\sqrt{\rho} / 2)$ maintains modest bursts size

E. Blanton and M. Allman, "On the impact of bursting on TCP performance," in *Passive and Active Measurement (PAM 2005) Lecture Notes in Comput. Science*. Springer, Berlin: vol. 3431, pp. 1–12, Mar. 2005.



Slow start sub-phases: pseudocode

```
if (cwnd < ssthresh)
{
  if ((cwnd <= ssthresh/4) && (flightsize < rwnd/4))
    set sub-phase = slow start sub-phase 1
  if ((cwnd > ssthresh/4) && (cwnd <= ssthresh/2) && (flightsize
  < rwnd/4))
    set sub-phase = slow start sub-phase 2
  if ((cwnd > ssthresh/4) && (flightsize >= rwnd/4) && (flightsize
  < rwnd/2))
    set sub-phase = slow start sub-phase 3
  if ((cwnd > ssthresh/2) && (flightsize >= rwnd/4) && (flightsize
  < rwnd/2))
    set sub-phase = slow start sub-phase 4
}
```



Adaptive *rwnd* increase mechanism

- Based on the ρ , *flightsize*, *cwnd* increment phase, and presence or absence of losses
- Compensates for long propagation delays when no losses occur
- Allows **one additional segment** (plus each first unacknowledged segment) to be sent when multiple losses have occurred in fast recovery phase
- Maintains the *rwnd* when losses occur and the TCP sender has exited the fast recovery phase

cwnd increment phase: slow start or congestion avoidance phase

Adaptive *rwnd* increase mechanism: pseudocode



```
if (flightsize < rwnd)
{
// no losses have occurred
if (snd_recover == 0)
    set rwnd to rwnd + rtt_dev_gain ×  $\rho$  × SMSS
// losses have occurred and in fast recovery phase
else if ((snd_una + SMSS ≤ snd_recover) &&
        (snd_recover != 0))
    set rwnd to rwnd + SMSS
else
    do nothing
}
```



Loss recovery mechanism

- Modifies the size of *cwnd* during the fast recovery phase based on:
 - current *cwnd*
 - number of acknowledged bytes
- Adds **200 ms** to the current time for computing the next RTO timer expiration to compensate for delayed ACK
- Limits the number of retransmissions from the retransmission buffer to **three segments** to prevent a large number of unnecessary or spurious retransmissions



Loss recovery mechanism: pseudocode

```
// in fast recovery phase
if (snd_una > snd_recover)
{
  if (cwnd <= acked_bytes)
    set cwnd to  $2 \times \text{SMSS}$ 
  else
    // deflate the congestion window by the number
    // of acknowledged data and add back two SMSS
    set cwnd to  $\textit{cwnd} - \textit{acked\_bytes} + (2 \times \text{SMSS})$ 
}
```



Roadmap

- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR) algorithm
- Performance evaluation of **TCP-ADaLR**:
 - simulation scenarios and results
 - fairness and friendliness scenarios
- Conclusions



Error model

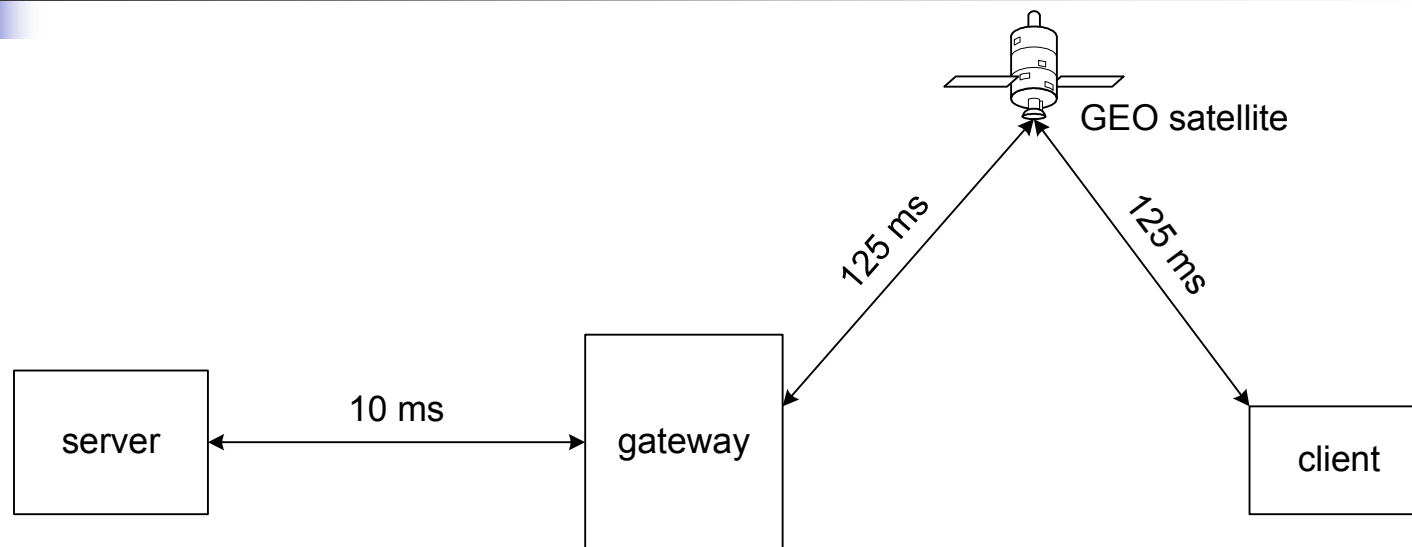
- GEO satellite link was modeled as an additive white Gaussian noise (AWGN) channel:
 - satellite client is a fixed user that has a line-of-sight (LoS) to the GEO satellite
 - satellite link exhibits random errors
 - $PER = 1 - (1 - BER)^N$, $N = 12,000$ bits
- Error correction threshold:
 - highest proportion of bit errors in a packet accepted by the receiver
 - equivalent to **PER** when the **BER is 10^{-10}**

PER: packet error rate

BER: bit error rate

N: number of bits in transmitted packet

Network topology



- Propagation delays are **one-way**
- Ethernet link between the server and the gateway is **full-duplex** with a **data rate of 10 Mb/s**
- GEO satellite link between the gateway and the client is asymmetric with **data rates of 2 Mb/s downlink** and **256 kb/s uplink**



Simulation scenarios and performance metrics



- Four scenarios with GEO satellite link:
 - ideal with **no losses**
 - ideal with **only congestion losses**
 - with **only error losses**
 - with **both congestion and error losses**
- Performance metrics:
 - FTP download response time
 - TCP goodput and throughput
 - satellite link throughput and utilization
 - HTTP page response time



Simulation scenarios and parameters

- TCP variants:
 - TCP-ADaLR SACK
 - TCP-ADaLR NewReno
 - TCP SACK
 - TCP NewReno
- Parameters:
 - FTP and HTTP applications with constant parameters
 - TCP parameters: standard OPNET TCP parameters with delayed ACK enabled and disabled



Simulated application parameters

HTTP webpage download application

Attribute	Value
HTTP specification	HTTP 1.1
Page inter-arrival time (s)	30
Main page object size (bytes)	10,710
Number of embedded objects	15
Embedded object size (bytes)	7,758
Simulated time (s)	1,000

FTP file download application

Attribute	Value
File inter-request time (s)	18,000
File size (MB)	50
Simulated time (hours)	5



TCP simulation parameters

TCP Parameter	Value
Initial RTO	3.0 s
Minimum RTO	1.0 s
Maximum RTO	64.0 s
Timer granularity	0.5 s
Persistent timeout	1.0 s
Maximum ACK delay	0.0 s
Maximum ACK segment	1
Duplicate ACK threshold	3
Sender maximum segment size (SMSS)	1,460 bytes
Slow start initial count	2
Receiver's advertised window (<i>rwnd</i>)	65,535 bytes
Retransmission threshold	6
RTT gain	0.125
RTT deviation gain	0.25
RTT deviation coefficient	4



Roadmap

- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR) algorithm
- Performance evaluation of **TCP-ADaLR**:
 - simulation scenarios with **no losses**
 - fairness and friendliness scenarios
- Conclusions



Scenario with no losses: HTTP page response time

TCP variant	Page response time (s)	
	With delayed ACK disabled	With delayed ACK enabled
TCP-ADaLR SACK	3.9	4.4
TCP-ADaLR NewReno	3.9	4.4
TCP SACK	4.3	4.9
TCP NewReno	4.3	4.9

- TCP-ADaLR variants show shorter page response time:
 - 10% with delayed ACK disabled
 - 9% with delayed ACK enabled
- Adaptive window increase mechanisms allow transmission of additional segments

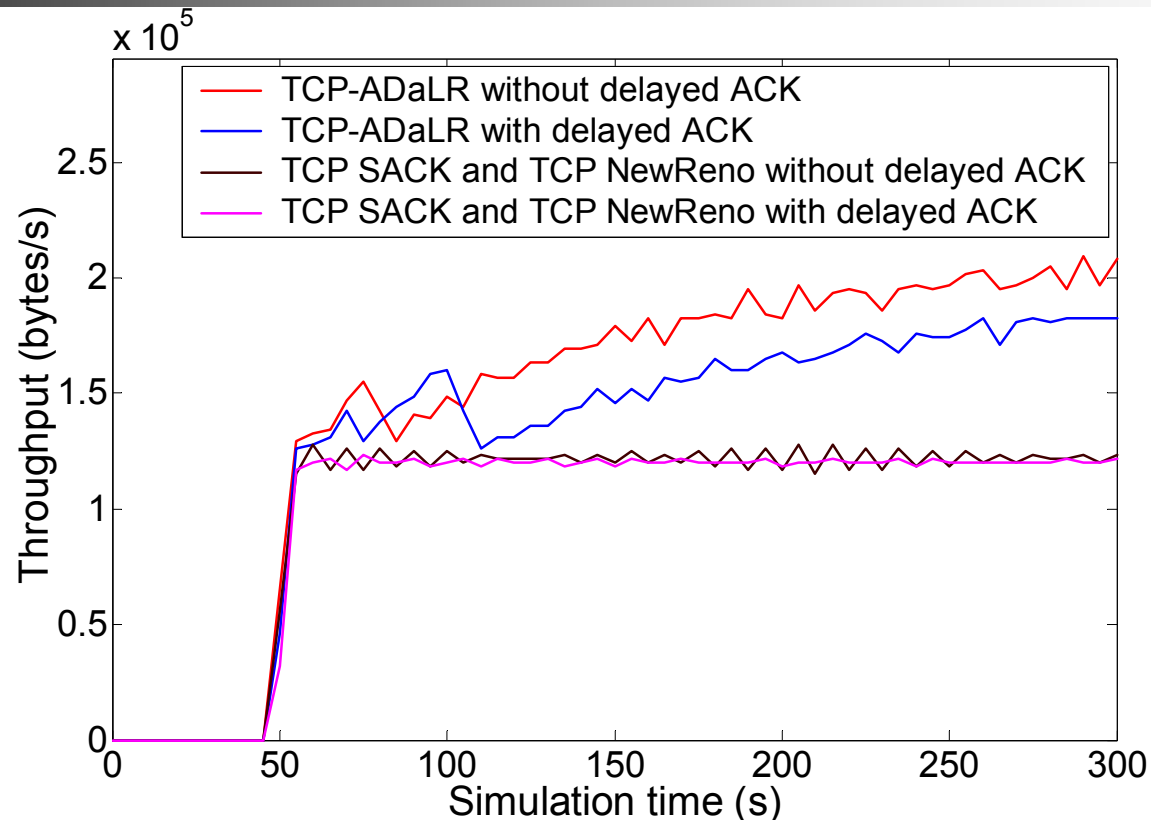
Scenario with no losses: FTP download response time



TCP variant	Download response time (s)	
	With delayed ACK disabled	With delayed ACK enabled
TCP-ADaLR SACK	333.4	360.6
TCP-ADaLR NewReno	333.4	360.6
TCP SACK	463.5	470.1
TCP NewReno	463.5	470.1

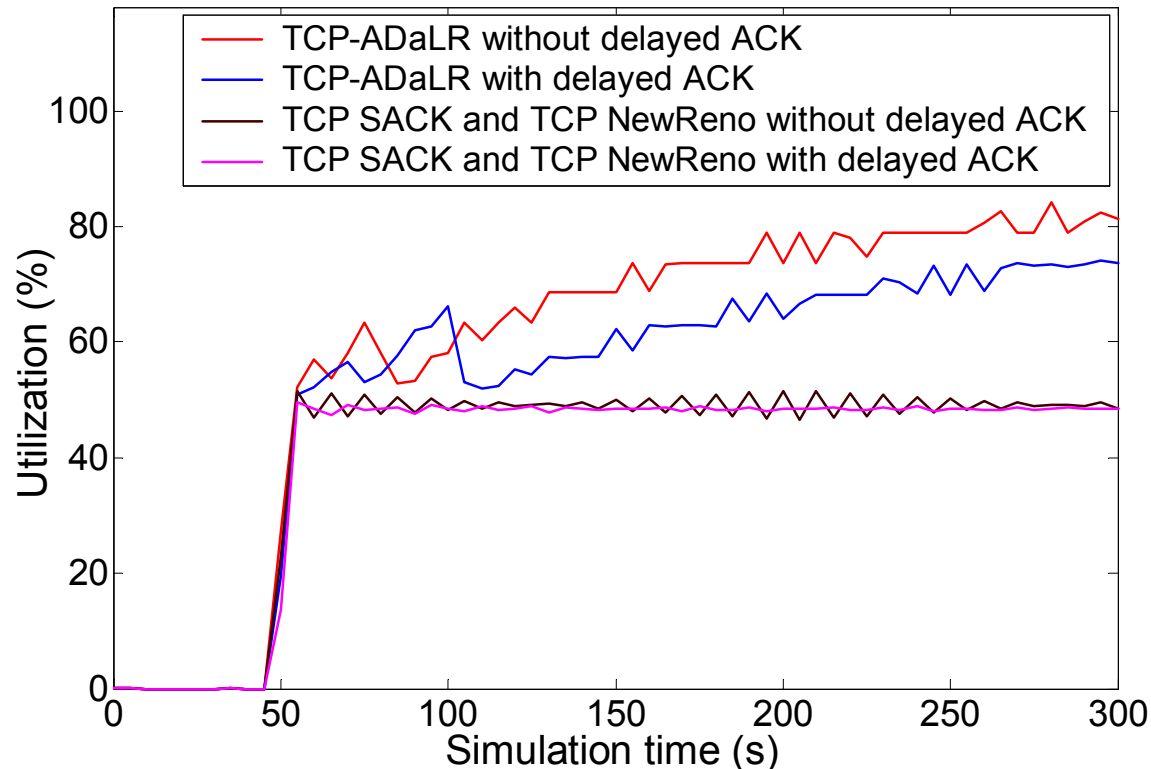
- TCP-ADaLR variants show shorter download response times:
 - 23% with delayed ACK disabled
 - 28% with delayed ACK enabled
- TCP-ADaLR algorithm does not degrade performance of TCP connections with delayed ACK disabled

Scenario with no losses: TCP throughput (FTP)



- **TCP-ADaLR** variants show higher TCP throughput than TCP SACK and TCP NewReno:
 - **63%** with delayed ACK disabled
 - **53%** with delayed ACK enabled

Scenario with no losses: satellite link utilization (FTP)



- **TCP-ADaLR** variants show higher satellite link utilization than TCP SACK and TCP NewReno:
 - **63%** with delayed ACK disabled
 - **61%** with delayed ACK enabled



Roadmap

- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR) algorithm
- Performance evaluation of **TCP-ADaLR**:
 - simulation scenario with **only congestion losses**
 - fairness and friendliness scenarios
- Conclusions

Scenario with only congestion losses: HTTP page response time



TCP variant	Page response time (s)	
	With delayed ACK disabled	With delayed ACK enabled
TCP-ADaLR SACK	10.3	11.0
TCP-ADaLR NewReno	11.1	11.0
TCP SACK	11.7	13.8
TCP NewReno	11.7	16.6

- TCP-ADaLR SACK exhibits shorter page response time than TCP SACK:
 - 12% with delayed ACK disabled
 - 33% with delayed ACK enabled
- Loss recovery mechanism enables quicker recovery from losses than TCP SACK and TCP NewReno

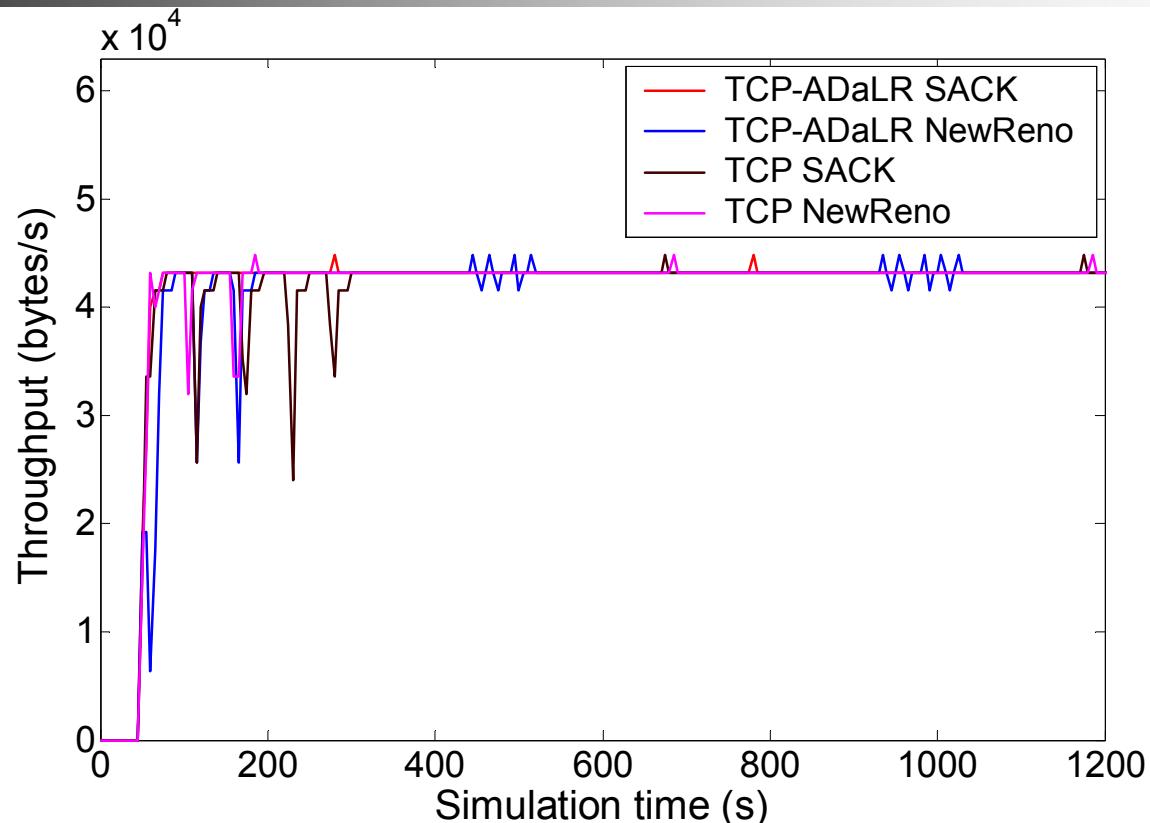
Scenario with only congestion losses: FTP download response time



TCP variant	Download response time (s)	
	With delayed ACK disabled	With delayed ACK enabled
TCP-ADaLR SACK	1,226.7	1,212.7
TCP-ADaLR NewReno	1, 232.4	1,228.0
TCP SACK	1, 226.7	1,224.8
TCP NewReno	1, 226.7	1,216.6

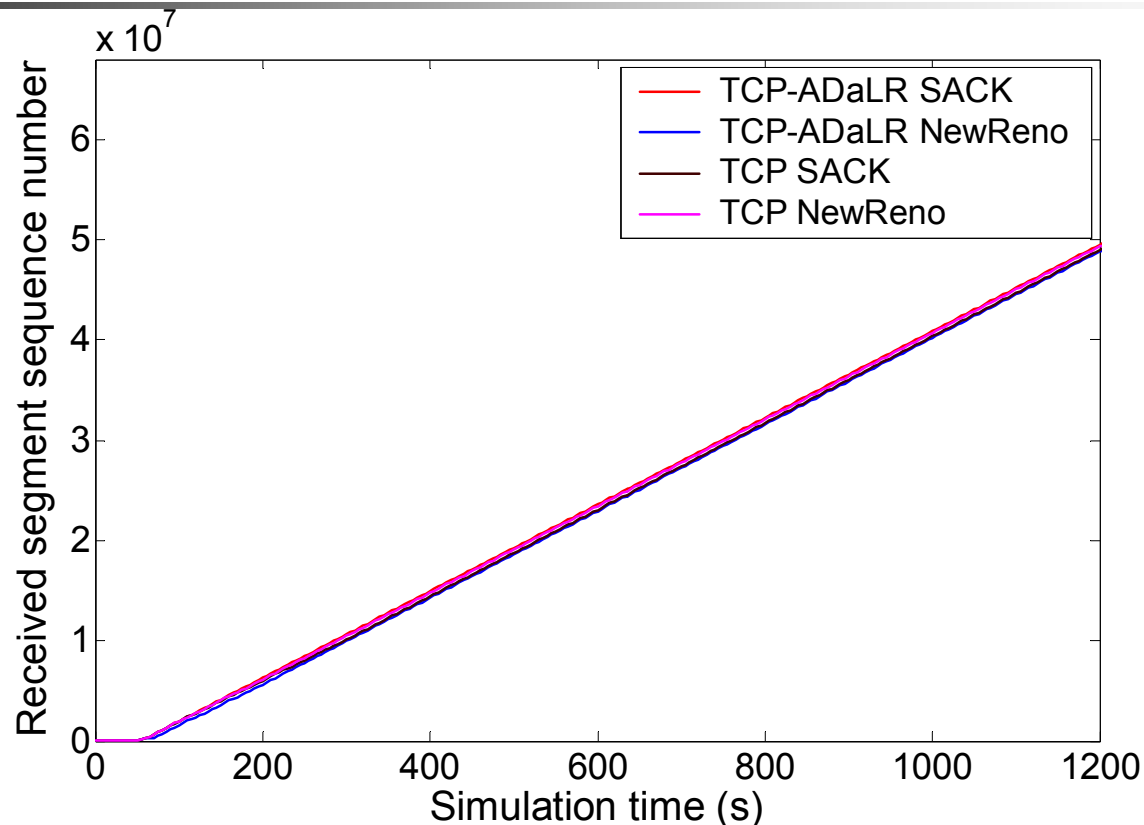
- TCP-ADaLR variants show download response times comparable to TCP SACK and TCP NewReno for the case with delayed ACK disabled
- TCP variants in the case with delayed ACK disabled exhibit similar performance as variants in the case with delayed ACK enabled

Scenario with only congestion losses: TCP throughput (FTP)



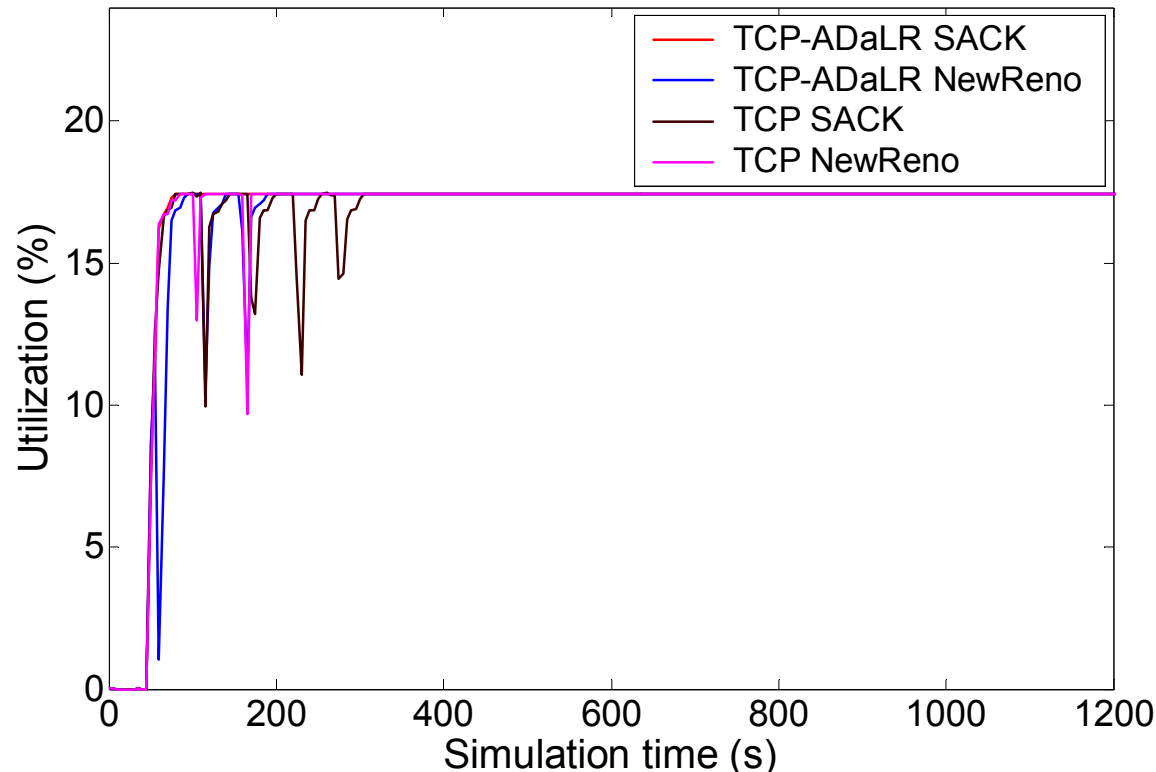
- **TCP-ADaLR** variants exhibit TCP throughput **comparable** to TCP SACK and TCP NewReno for the case **with delayed ACK enabled**
- Performance degradation of the four TCP variants reflects the impact of congestion

Scenario with only congestion losses: TCP goodput (FTP)



- **TCP-ADaLR** variants exhibit TCP goodput **comparable** to TCP SACK and TCP NewReno for the case **with delayed ACK enabled**
- Received segment sequence number is used as indicator of goodput

Scenario with only congestion losses: satellite link utilization (FTP)



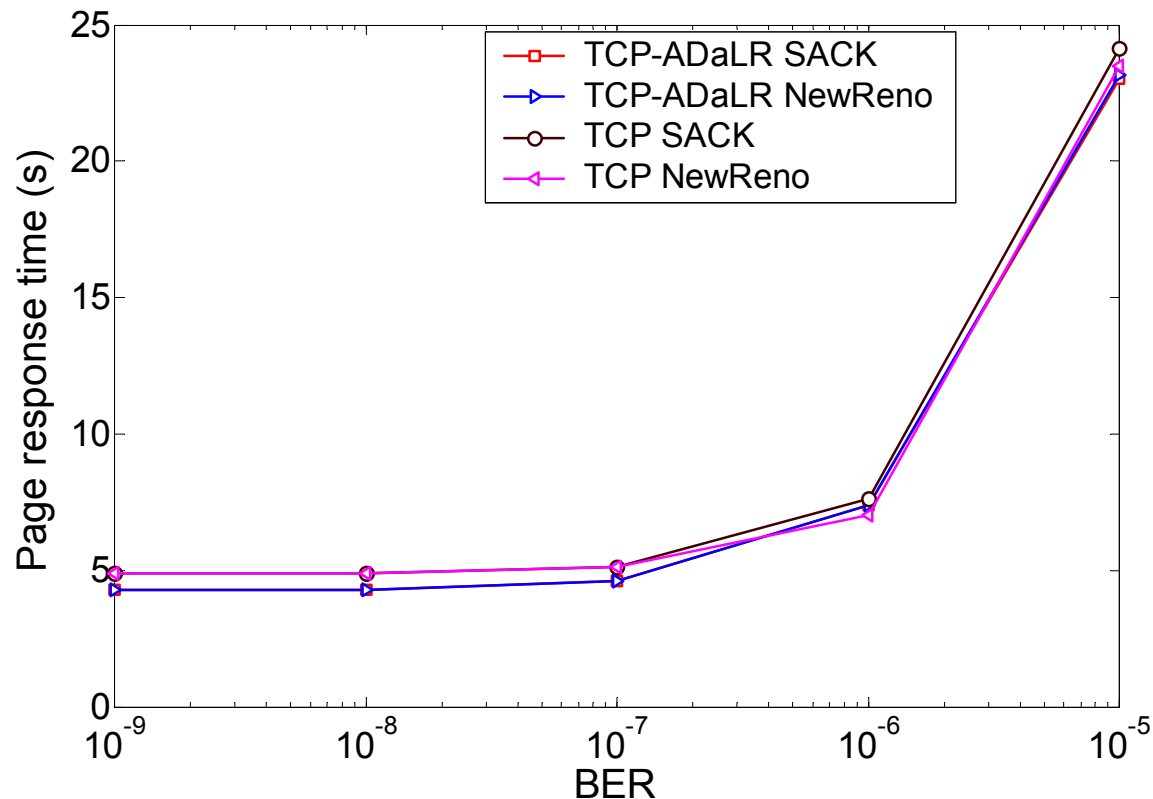
- TCP-ADaLR variants exhibit satellite link utilization comparable to TCP SACK and TCP NewReno for the case with delayed ACK enabled
- Satellite link utilization exhibits drops when congestion losses are detected



Roadmap

- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR) algorithm
- Performance evaluation of **TCP-ADaLR**:
 - simulation scenario with **only error losses**
 - fairness and friendliness scenarios
- Conclusions

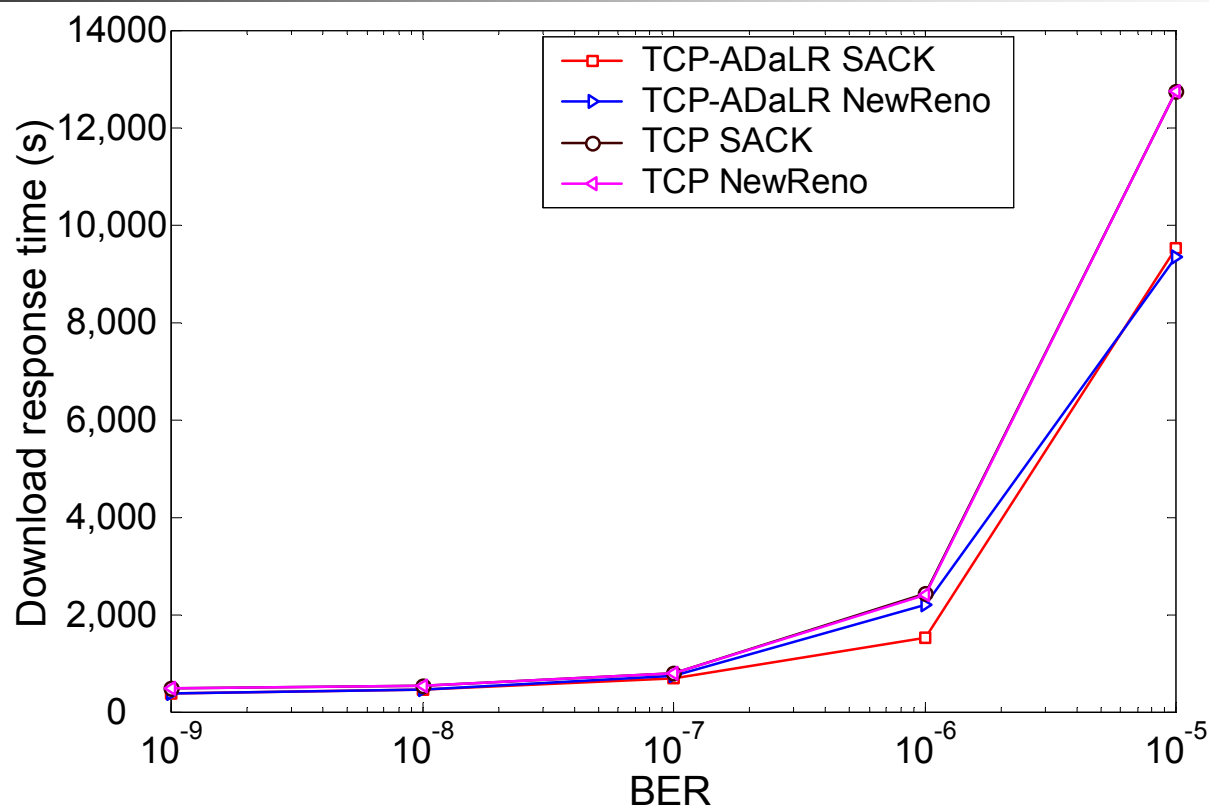
Scenario with only error losses: HTTP page response time



- **TCP-ADaLR SACK** exhibits **2%–12%** shorter page response time than TCP SACK for the case **with delayed ACK enabled**
- **TCP-ADaLR NewReno** exhibits **4%–12%** shorter page response time than TCP NewReno for the case **with delayed ACK enabled**

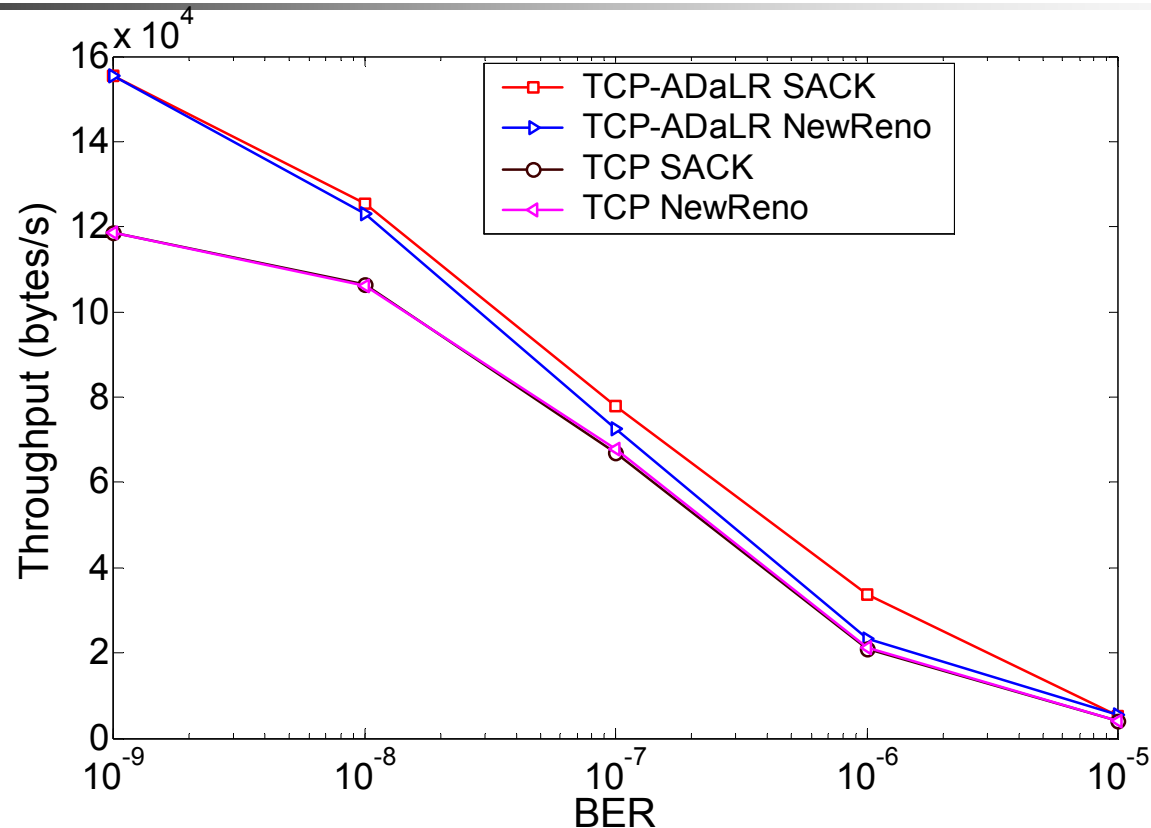


Scenario with only error losses: FTP download response time



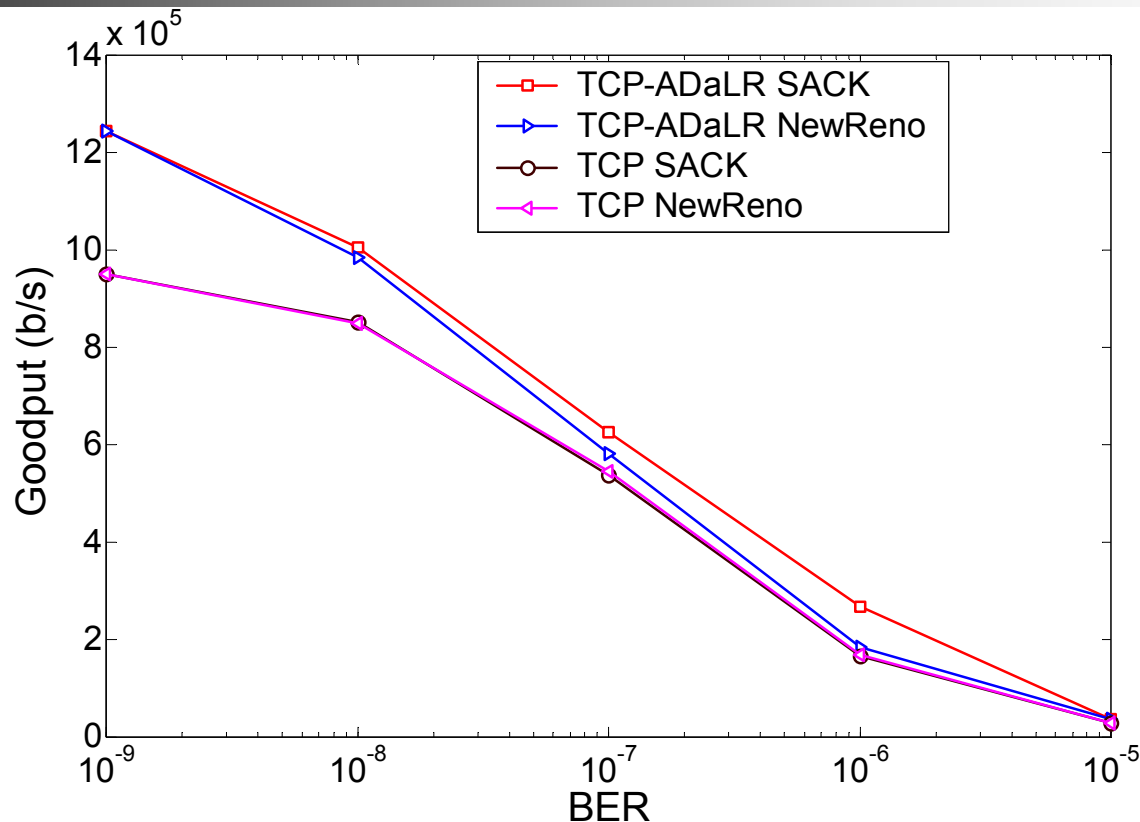
- **TCP-ADaLR SACK** exhibits **13%–37%** shorter download response time than TCP SACK for the case with delayed ACK enabled
- **TCP-ADaLR NewReno** exhibits **6%–26%** shorter download response times than TCP NewReno

Scenario with only error losses: TCP throughput (FTP)



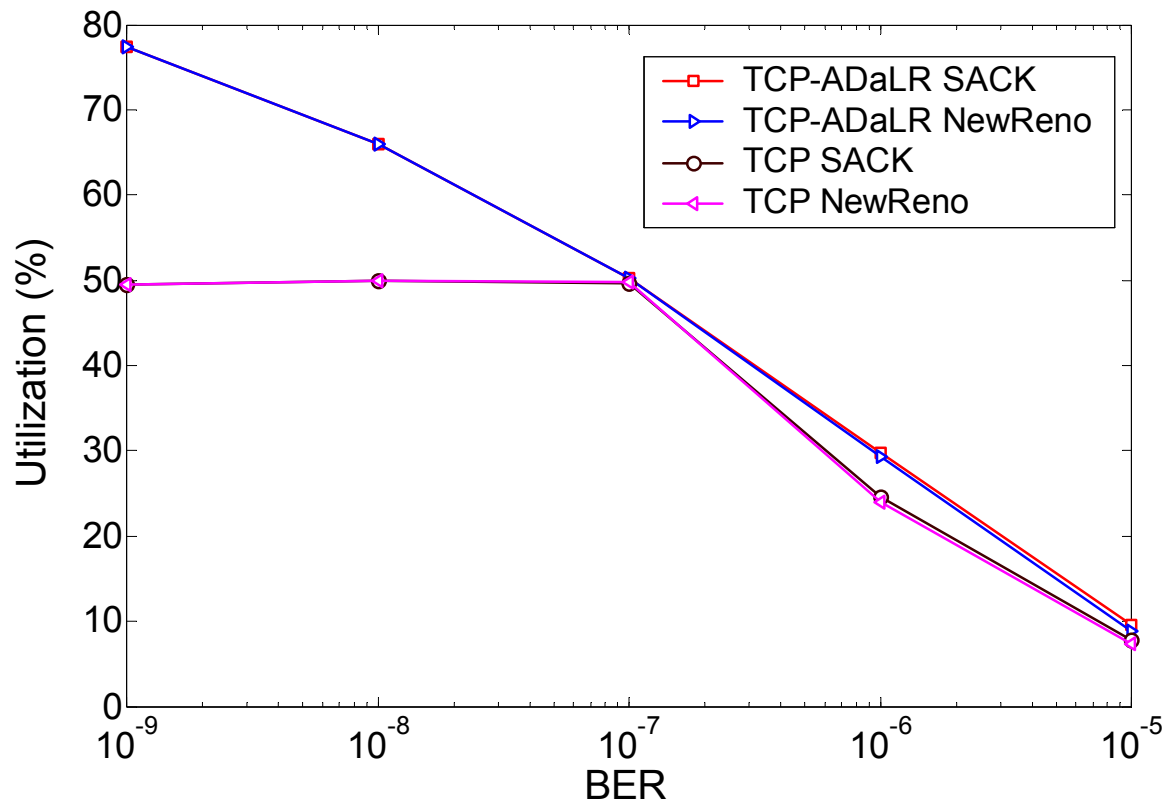
- **TCP-ADaLR SACK** exhibits **16%–61%** higher TCP throughput than TCP SACK for the case **with delayed ACK enabled**
- **TCP-ADaLR NewReno** exhibits **6%–36%** higher TCP throughput than TCP NewReno for the case **with delayed ACK enabled**

Scenario with only error losses: TCP goodput (FTP)



- **TCP-ADaLR SACK** exhibits **16%–61%** higher goodput than TCP SACK for the case **with delayed ACK enabled**
- **TCP-ADaLR NewReno** exhibits **6%–34%** higher goodput than TCP NewReno for the case **with delayed ACK enabled**

Scenario with only error losses: satellite link utilization (FTP)



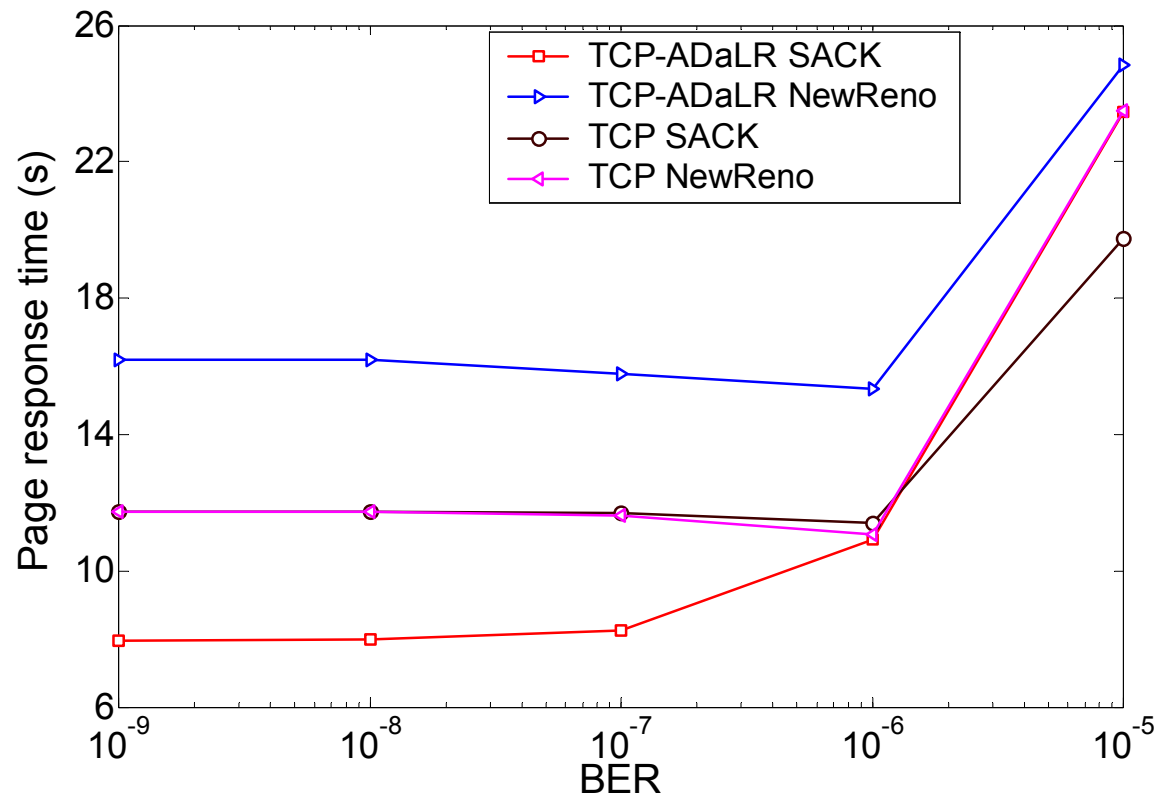
- **TCP-ADaLR SACK** exhibits **2%–27%** higher satellite link utilization than TCP SACK for the case **with delayed ACK enabled**
- **TCP-ADaLR NewReno** exhibits **1%–27%** higher satellite link utilization than TCP NewReno



Roadmap

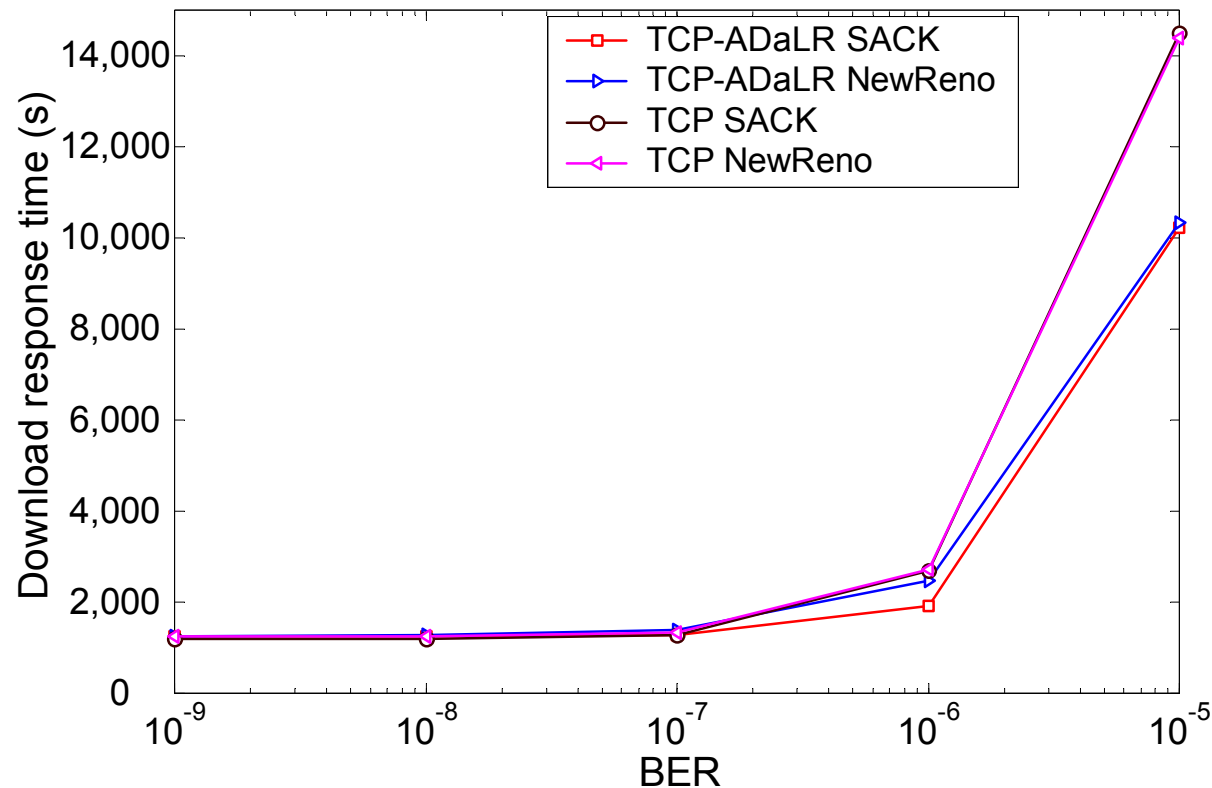
- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR) algorithm
- Performance evaluation of **TCP-ADaLR**:
 - simulation scenario with **both congestion and error losses**
 - fairness and friendliness scenarios
- Conclusions

Scenario with both congestion and error losses: HTTP page response time



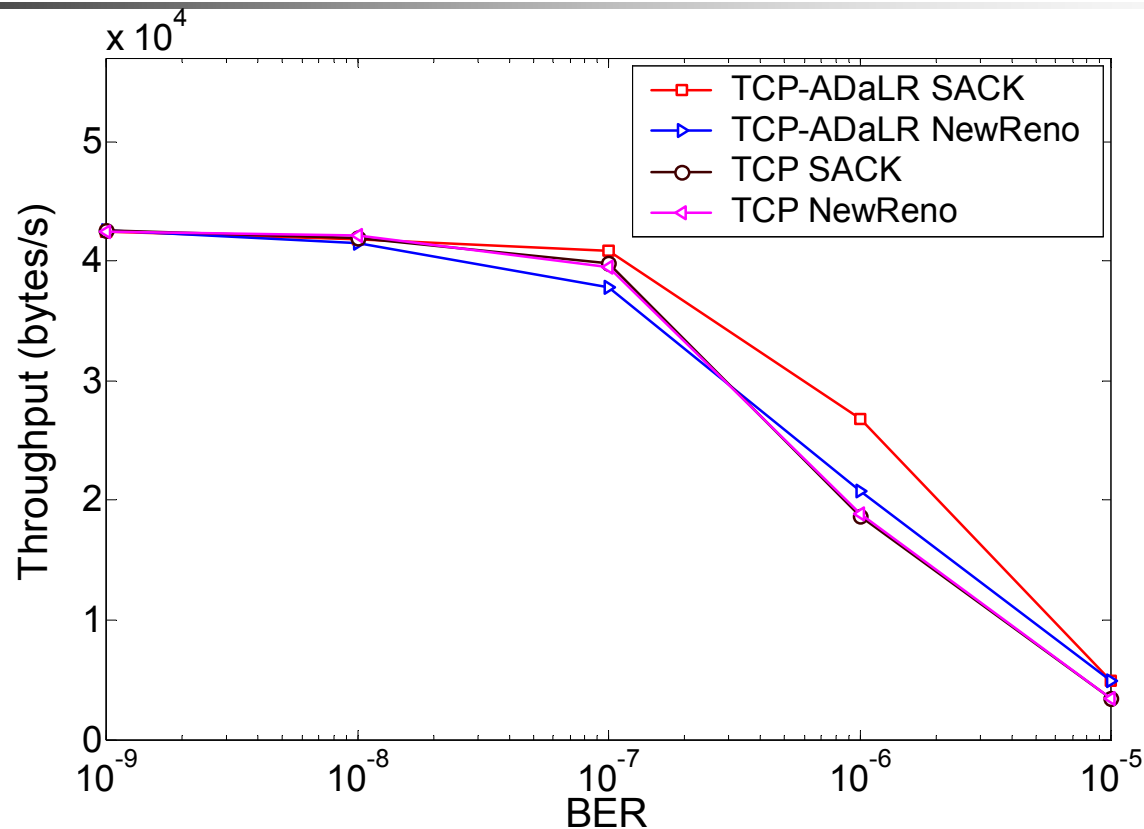
- **TCP-ADaLR SACK** exhibits up to **32%** shorter page response time than TCP SACK for the case **with delayed ACK enabled**
- TCP-ADaLR NewReno worse performance may be due to loss of original and retransmitted segments

Scenario with both congestion and error losses: FTP download response time



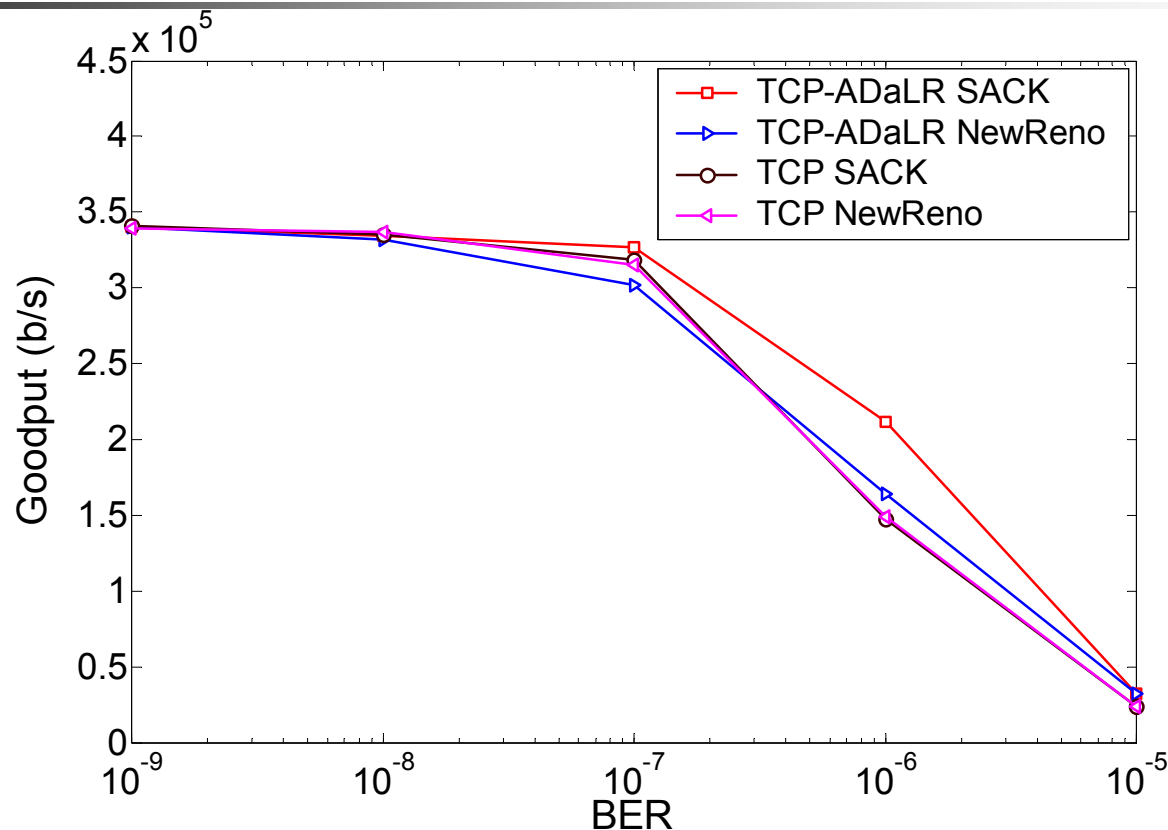
- **TCP-ADaLR SACK** exhibits **28%–29%** lower download response times than TCP SACK for the case with delayed ACK enabled
- **TCP-ADaLR NewReno** exhibits **9%–29%** lower download response times than TCP NewReno

Scenario with both congestion and error losses: TCP throughput (FTP)



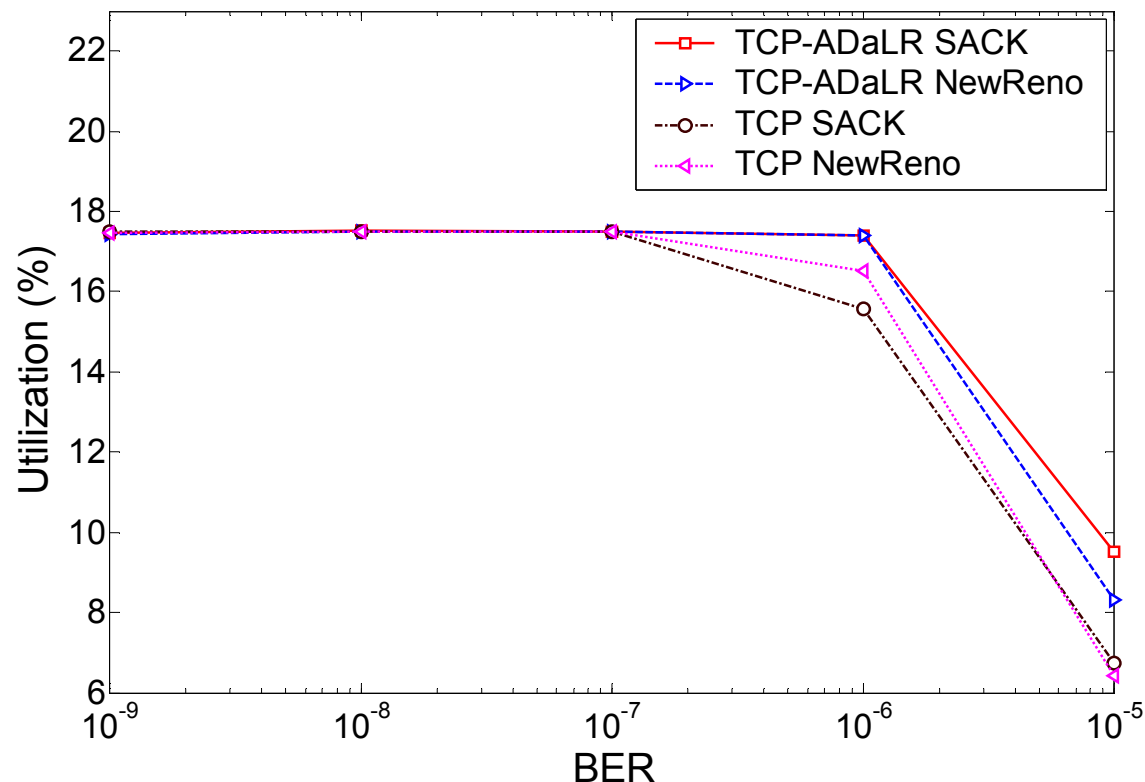
- **TCP-ADaLR SACK** exhibits **42%–43%** higher TCP throughput than TCP SACK for the case with delayed ACK enabled
- **TCP-ADaLR NewReno** exhibits **10%–39%** higher TCP throughput than TCP NewReno

Scenario with both congestion and error losses: TCP goodput (FTP)



- **TCP-ADaLR SACK** exhibits **36%–43%** higher TCP goodput than TCP SACK for the case with delayed ACK enabled
- **TCP-ADaLR NewReno** exhibits **10%–31%** higher TCP goodput than TCP NewReno

Scenario with both congestion and error losses: satellite link utilization (FTP)



- **TCP-ADaLR SACK** exhibits **11%–41%** higher satellite link utilization than TCP SACK for the case **with delayed ACK enabled**
- **TCP-ADaLR NewReno** exhibits **5%–29%** higher satellite link utilization than TCP NewReno



Roadmap

- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR) algorithm
- Performance evaluation of **TCP-ADaLR**:
 - simulation scenarios and results
 - **fairness and friendliness scenarios**
- Conclusions



Fairness and friendliness

- TCP connections in deployed networks have:
 - identical or distinct **RTTs**
 - identical or distinct **TCP variants**
 - coexist and share bottleneck links
- **Fairness**: coexisting connections with identical TCP variants achieve equal bandwidth allocation
- **Friendliness**: coexisting identical TCP connections with distinct TCP variants achieve equal bandwidth allocation



Fairness and friendliness

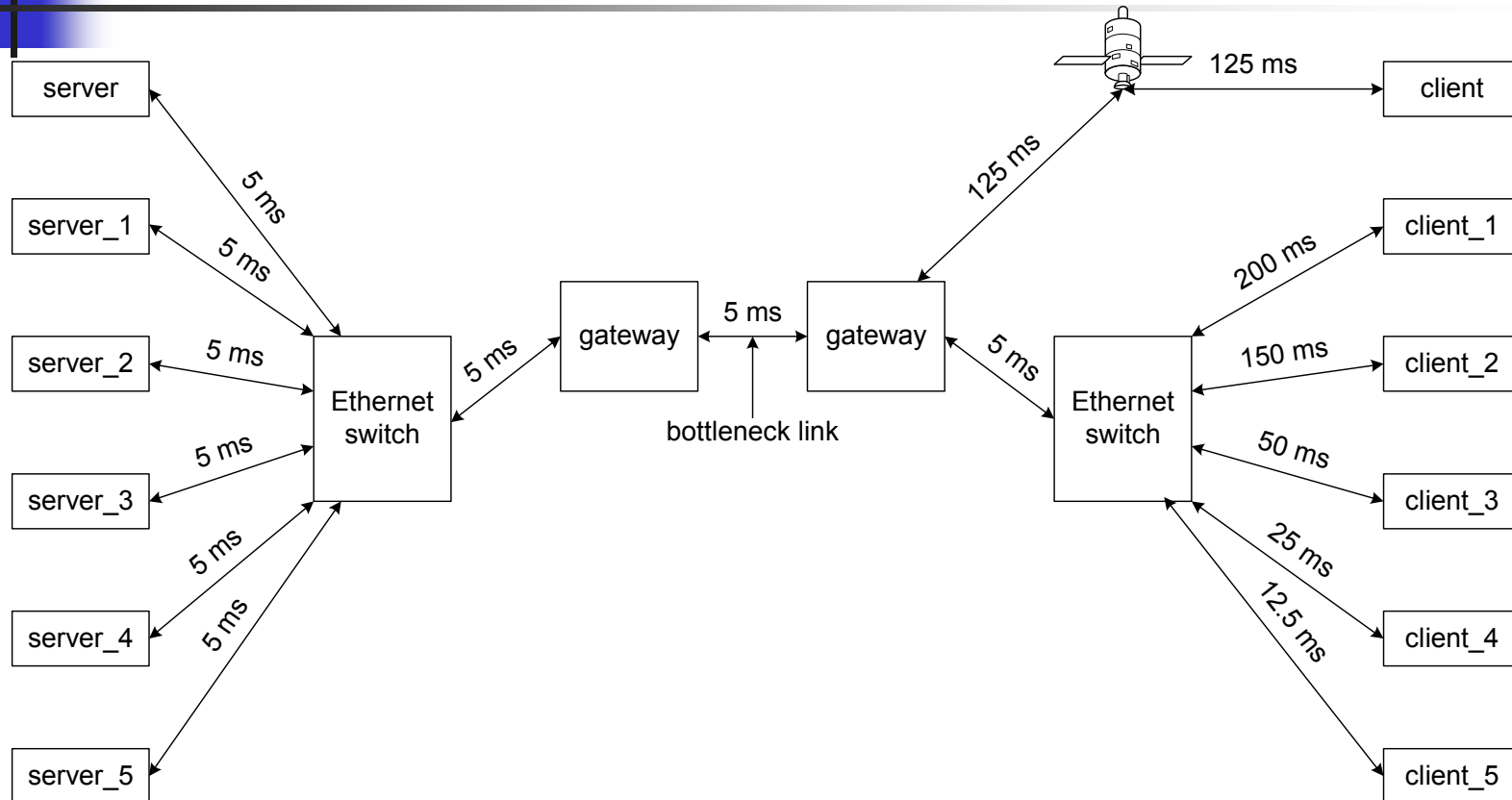
- Jain's metric of fairness:

$$FI = \frac{(\sum_{j=1}^n t_j)^2}{n \times (\sum_{j=1}^n t_j^2)}$$

- FI is fairness (friendliness) index
- n is the number of competing connections
- t_j is the average throughput of the j th connection
- $1/n \leq FI \leq 1$: $1/n$ corresponds to unfair and 1 corresponds to fair bandwidth allocation

D. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks," *J. of Comput. Netw. and ISDN Syst.*, vol. 17, no. 1, pp. 1-14, June 1989.

Fairness and friendliness: network topology



- propagation delays are **one-way**
- links are **full-duplex** with **10 Mb/s data rate**



Fairness and friendliness scenarios

- Fairness scenarios with **six** coexisting TCP connections with six distinct RTTs:
 - 25 ms, 50 ms, 100 ms 300 ms, 400 ms, and 500 ms
 - TCP-ADaLR NewReno: **TCP-ADaLR SACK** and **TCP-ADaLR NewReno** exhibit identical performance in an ideal satellite link without losses
 - TCP NewReno
- Friendliness scenario with **six** coexisting identical TCP connections:
 - RTT = 25 ms
 - 3 **TCP-ADaLR NewReno** RTT connections
 - 3 TCP NewReno RTT connections:



TCP throughput: fairness scenario

TCP throughput achieved by six competing TCPADaLR and TCP NewReno connections

RTT (ms)	Average throughput (bytes/s)	
	TCP-ADaLR	TCP NewReno
25	283,404.6	322,418.0
50	281,750.6	300,629.1
100	268,984.6	263,129.2
300	195,099.8	158,601.5
400	175,343.8	129,560.8
500	160,897.4	109,239.5



TCP throughput: friendliness scenario

Average throughput achieved by six identical competing connections using distinct TCP variants. RTT is set to 25 ms

TCP variant	Average throughput (bytes/s)
TCP-ADaLR	354,451.6
TCP-ADaLR	356,565.9
TCP-ADaLR	356,906.8
TCP NewReno	352,012.7
TCP NewReno	351,913.3
TCP NewReno	351,748.1



Fairness (friendliness) indices

TCP variant	Fairness index
TCP-ADaLR NewReno	0.9510
TCP NewReno	0.8650

- TCP-ADaLR NewReno exhibits higher a **higher fairness index** than TCP NewReno
- Connections with longer RTTs have fair share without starving shorter RTT connections

TCP variant	Friendliness index
TCP-ADaLR NewReno and TCP NewReno	0.99996

- TCP-ADaLR NewReno is friendly to TCP NewReno connections
- TCP NewReno connections have fair share of the bottleneck link's capacity

A decorative graphic on the left side of the slide, featuring overlapping yellow, red, and blue squares with a black crosshair.

Roadmap

- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR) algorithm
- Performance evaluation of TCP-ADaLR:
 - simulation scenarios and results
 - fairness and friendliness scenarios
- **Conclusions**



Conclusions

- **TCP-ADaLR SACK** and **TCP-ADaLR NewReno** perform better than TCP SACK and TCP NewReno for both cases **with** and **without delayed ACK** in:
 - absence of congestion and error losses
 - presence of error losses
 - presence of both congestion and error losses
- **TCP-ADaLR SACK** and **TCP-ADaLR NewReno** perform comparably to TCP NewReno and TCP SACK in the presence of congestion
- **TCP-ADaLR SACK** exhibits the overall best performance
- **TCP-ADaLR** algorithm does not degrade performance of TCP connections **without delayed ACK**



Conclusions

- Deployment of **TCP-ADaLR** in existing networks:
 - requires modifications only at the TCP sender with minimal:
 - processing overhead (computation of **scaling component ρ**)
 - memory overhead
 - preserves TCP end-to-end semantics
 - is compatible with IP security for IP payload encryption and authentication
- **TCP-ADaLR** ensures fair capacity allocation for coexisting connections at the bottleneck link
- **TCP-ADaLR** is friendly to TCP NewReno



References

- D. E. Comer, *Internetworking with TCP/IP: Principles, Protocols, and Architecture*. vol. 1. Upper Saddle River, NJ: Prentice Hall, 2000, pp. 209–218
- B. R. Elbert, *The Satellite Communications Applications Handbook*, 2nd ed. Norwood, MA: Artech House, 2004.
- M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," *IETF RFC 2581*, Apr. 1999.
- S. Floyd, T. Henderson, and A. Gurtov, "The NewReno modification to TCP's fast recovery algorithm," *IETF RFC 3782*, Apr. 2004.
- M. Fomenkov, K. Keys, D. Moore, and K. Claffy, "Longitudinal study of Internet traffic in 1998-2003," in *Proc. ACM Winter Int. Symp. Inf. and Commun. Technologies*, Cancun, Mexico, Jan. 2004, pp. 1–6.
- C. Fraleigh et al., "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Network*, vol. 17, no. 6, pp. 6–16, Nov./Dec. 2003.
- S Subramanian, S. Sivakumar, W. J. Phillips, and W. Robertson, "Investigating TCP performance issues in satellite networks," in *Proc. Third IEEE Commun. Netw. and Services Research Conf.*, Halifax, NS, Canada, May 2005, pp. 327–332.
- J. Sing and B. Soh, "TCP performance over geostationary satellite links: problems and solutions," in *Proc. 12th IEEE Int. Conf. on Netw.*, Guadeloupe, French Caribbean, Nov. 2004, vol. 1, pp. 14–18.



References

- I. F. Akyildiz, G. Morabito, and S. Palazzo, "TCP-Peach: a new congestion control scheme for satellite IP networks," *IEEE/ACM Trans. Netw.*, vol. 9, no. 3, pp. 307–321, June 2001.
- H. Obata, K. Ishida, S. Takeuchi, and S. Hanasaki, "TCP-STAR: TCP Congestion Control Method for Satellite Internet" in *IEICE Trans. Commun.*, vol. E89-B, no. 6, pp. 1766–1773, June 2006.
- C. Caini and R. Firrincieli, "TCP Hybla: a TCP enhancement for heterogeneous networks," *Int. J. Satellite Commun. Netw.*, vol. 22, no. 5, pp. 547–566, Sept. 2004.
- J. Sing and B. Soh, "TCP New Vegas: improving the performance of TCP Vegas over high latency links," in *Proc. Fourth IEEE Int. Symp. on Netw. Comput. and Appl.*, Cambridge, MA, July 2005, pp. 73–82.
- J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance enhancing proxies intended to mitigate link-related degradations," *RFC 3135*, June 2001.
- Y. Shang and M. Hadjithodosiou, "TCP splitting protocol for broadband and aeronautical satellite network," in *Proc. 23rd IEEE Digital Avionics Syst. Conf.*, Salt Lake City, UT, Oct. 2004, vol. 2, pp. 11.C.3-1–11.C.3-9.
- C. Caini, R. Firrincieli, and D. Lacamera, "PEPsal: a performance enhancing proxy designed for TCP satellite connections," in *Proc. 63rd IEEE Veh. Technol. Conf.*, Melbourne, Australia, Feb. 2006, vol. 6, pp. 2607–2611.
- G. Ciccarese, M. De Blas, L. Patrono, P. Marra, and G. Tomasicchio, "An IPSec-aware TCP PEP for integrated mobile satellite networks," in *Proc. 15th IEEE Int. Symp. on Pers., Indoor and Mobile Radio Commun. (PIMRC 2004)*, Barcelona, Spain, Sept. 2004, vol. 4, pp. 2362–2366.



References

- E. A. Faulkner, A. P. Worthen, J. B. Schodorf, and J. D. Choi, "Interactions between TCP and link layer protocols on mobile satellite links," in *Proc. IEEE MILCOM*, Monterey, CA, Nov. 2004, vol. 1, pp. 535–541.
- J. Sing and B. Soh, "On the use of snoop with geostationary satellite links," in *Proc. Third IEEE Int. Conf. on Inf. Technol. and Appl. (ICITA 2005)*, Sydney, Australia, July 2005, vol. 2, pp. 689–694.
- R. Wang, V. Bandekodige, and M. Banerjee, "An experimental evaluation of link delay impact on throughput performance of TCP and SCPS-TP in space communications," in *Proc. 60th IEEE Veh. Technol. Conf.*, Los Angeles, CA, Sept. 2004, vol. 6, pp. 4061–4065.
- K. Zhou, K. L. Yeung, and V. O. K. Li, "P-XCP: a transport layer protocol for satellite IP networks," in *Proc. IEEE GLOBECOM*, Dallas, TX, Dec. 2004, vol. 5, pp. 2707–2711.
- M. Allman, "TCP congestion control with appropriate byte counting (ABC)," *IETF RFC 3465*, Feb. 2003.
- E. Blanton and M. Allman, "On the impact of bursting on TCP performance," in *Passive and Active Measurement (PAM 2005) Lecture Notes in Comput. Science*. Springer, Berlin: vol. 3431, pp. 1–12, Mar. 2005.
- D. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks," *J. of Comput. Netw. and ISDN Syst.*, vol. 17, no. 1, pp. 1-14, June 1989.
- OPNET Modeler software [Online]. Available: <http://www.opnet.com/products/modeler/home.html> .