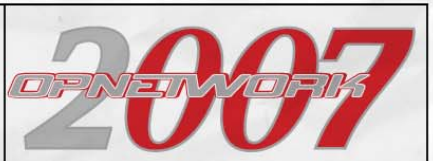
Deliver To:
OPNETWORK 2007
Ronald Reagan Building
Washington, D.C.

OPNET Model of TCP with adaptive delay and loss response for broadband GEO satellite 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

Roadmap



- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR):
  - algorithm description
  - OPNET implementation
- 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

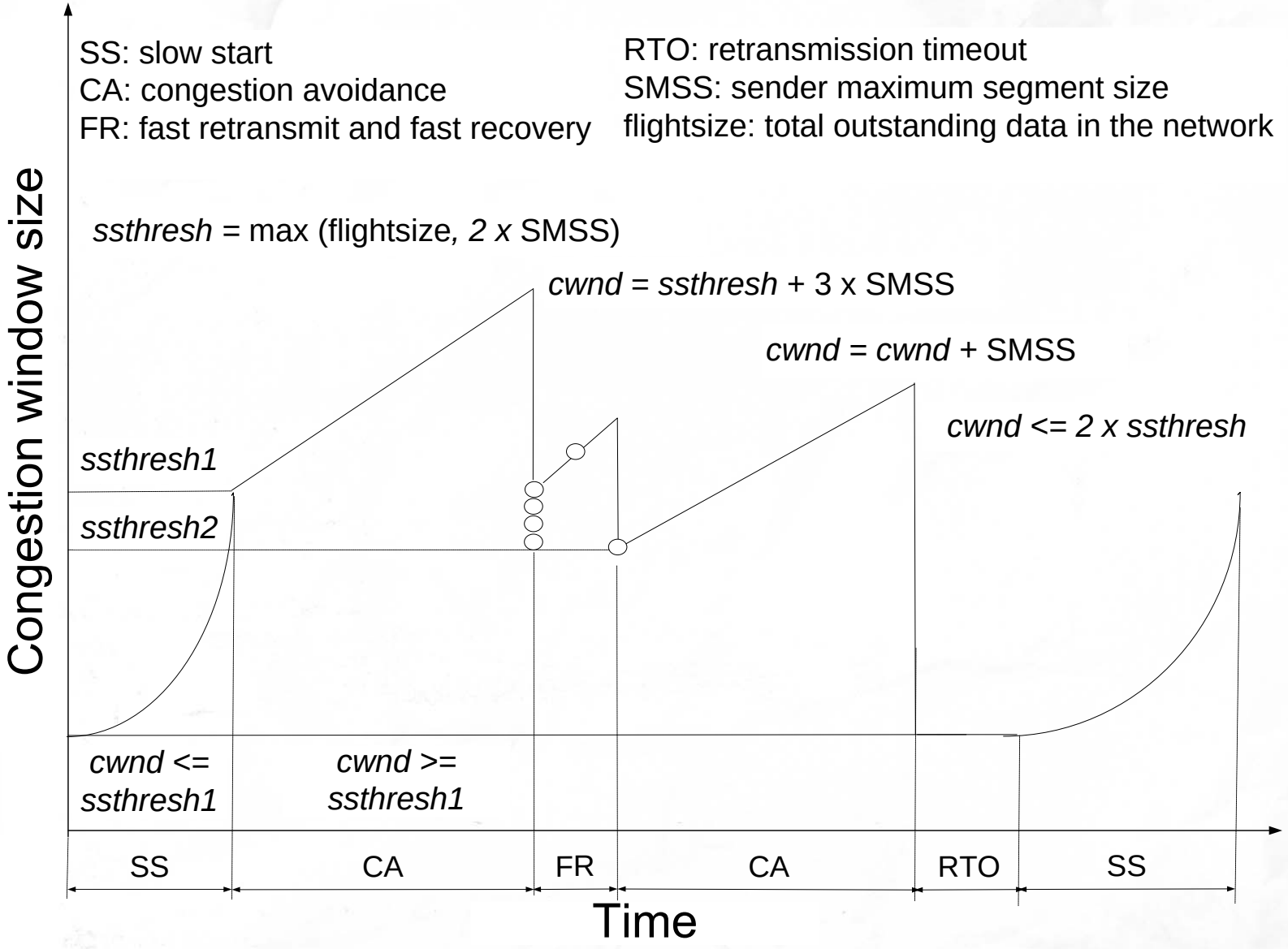
Roadmap

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

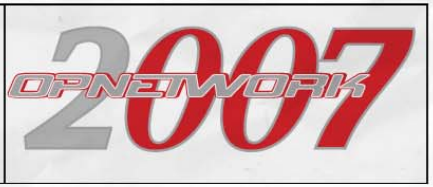
TCP congestion control algorithms

SS: slow start
CA: congestion avoidance
FR: fast retransmit and fast recovery

RTO: retransmission timeout
SMSS: sender maximum segment size
flightsize: total outstanding data in the network



TCP delayed acknowledgement option



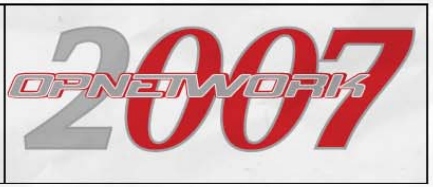
- Allows TCP receivers to send an acknowledgement (ACK) for every second consecutive full-sized segment received
- Implemented by many TCP receivers in the Internet:
  - default interval period: 200 ms
  - maximum interval period: 500 ms
- Reduces protocol processing overhead
- Increases network efficiency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

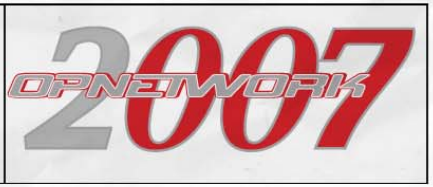
A full-sized segment is equivalent to the sender maximum segment size (SMSS)

Broadband GEO satellite networks



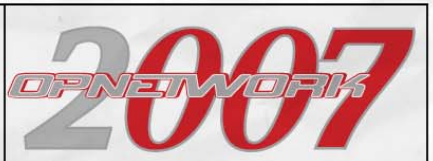
- Transmit and receive data using frequencies relayed by geostationary earth orbit (GEO) satellites
- Provide global Internet services for areas with limited or no terrestrial cable infrastructure
- Offer high data rates of the order of 1 Mb/s or higher through high-bandwidth GEO satellite links
- Employ GEO satellite links characterized by:
  - high bit error rates
  - long propagation delays
  - path asymmetry (uplink and downlink bandwidth)

TCP and GEO satellite links



- TCP performance in broadband networks employing GEO satellite links needs improvement:
  - packet losses occur in satellite networks due to GEO satellite link characteristics
  - packet losses are misinterpreted as congestion indication
  - *cwnd* is reduced leading to TCP performance degradation
- TCP connections with delayed ACK in GEO satellite links:
  - exhibit degraded performance than TCP connections without delayed ACK in the presence of errors
  - show underutilization of GEO satellite link capacity during the TCP slow start phase

Related work

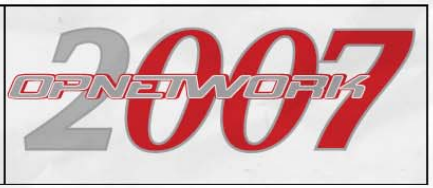


- End-to-end solutions preserve the end-to-end semantics of TCP: TCP-Peach, TCP-Hybla
- Split connection solutions violate end-to-end semantics of TCP
- Link layer solutions employ FEC and ARQ techniques for detecting and retransmitting lost segments at the link layer
- Non-TCP satellite-optimized solutions employ standard TCP algorithms and/or satellite specific algorithms for use in satellite segments of split TCP connections

Roadmap

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

TCP-ADaLR: TCP with adaptive delay and loss response



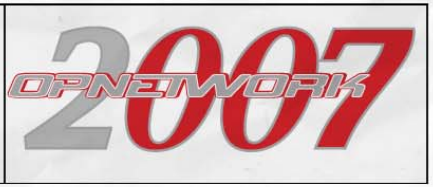
- End-to-end solution for improving TCP performance in broadband GEO satellite networks:
  - scaling component ρ
  - adaptive *cwnd* increase mechanism
  - adaptive *rwnd* increase mechanism
  - loss recovery mechanism
- Requires modifications only at the TCP sender
- Incorporates the delayed ACK option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

Scaling component ρ

- Used to increase the *cwnd* increment during the slow start and congestion avoidance phases
- Calculated as:
  - ρ = (*sampleRTT s/1 s*) × *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



- Based on the presence or absence of losses and ρ
- Slow start phase is divided into four sub-phases based on current *cwnd* and the *flightsize*, *cwnd* is incremented by:
  - $(\sqrt{\rho} / 4)$ x SMSS if no losses have occurred and the value of $\rho \geq 15$
  - SMSS if losses have occurred as in conventional TCP
- Congestion avoidance phase increment *cwnd* by $(\sqrt{\rho} / 2)$ x SMSS x SMSS / *cwnd* if $\rho \geq 15$:
  - losses have occurred and TCP sender is out of fast recovery
  - *flightsize* is less than (*rwnd* / 2)

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 ≥ 250 ms
- Selected based on simulation result of an FTP file download for various RTTs
- $(\sqrt{\rho} / 4) \times SMSS$ is equivalent to a value between (1 - 2) x SMSS and prevents large line-rate bursts
- $(\sqrt{\rho} / 2)$ maintains modest 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

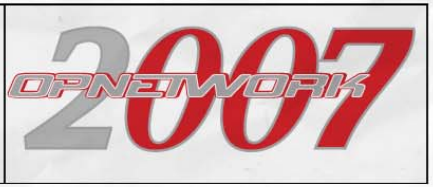
M. Allman, "TCP congestion control with appropriate byte counting (ABC)," *IETF RFC 3465*, Feb. 2003.

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



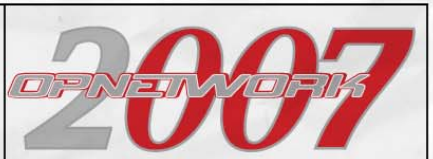
- Based on the ρ, *flightsize*, *cwnd* increment phase, and presence or absence of losses
- Compensates for long propagation delays when no losses have occurred
- Allows one additional segment (plus each first unacknowledged segment) to be sent when multiple losses have occurred in fast recovery phase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

cwnd increment phase: slow start or congestion avoidance phase

Adaptive *rwnd* increase mechanism: pseudocode

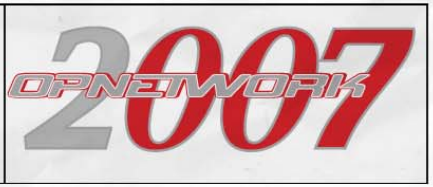
```
if (flightsize < rwnd)
        {
        // no losses have occurred
        if (snd_recover = = 0)
                set rwnd to rwnd + rtt_dev_gain × ρ × SMSS
        // losses have occurred and in fast recovery phase
        else if ((snd_una + SMSS <= snd_recover) &&
                (snd_recover != 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 × SMSS
     else
         // deflate the congestion window by the number
         // of acknowledged data and add back two SMSS
         set cwnd to cwnd - acked_bytes + (2  × SMSS)
    }
```

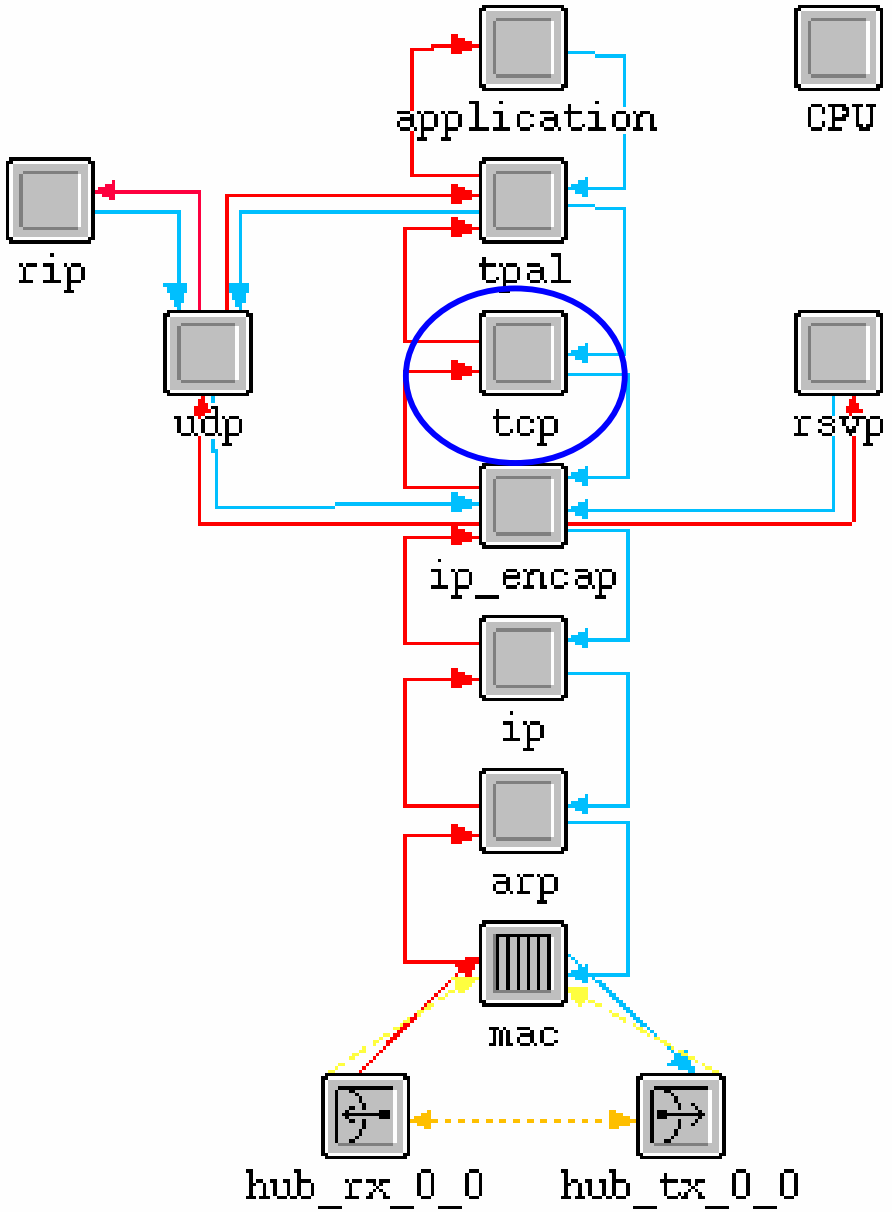
Roadmap

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

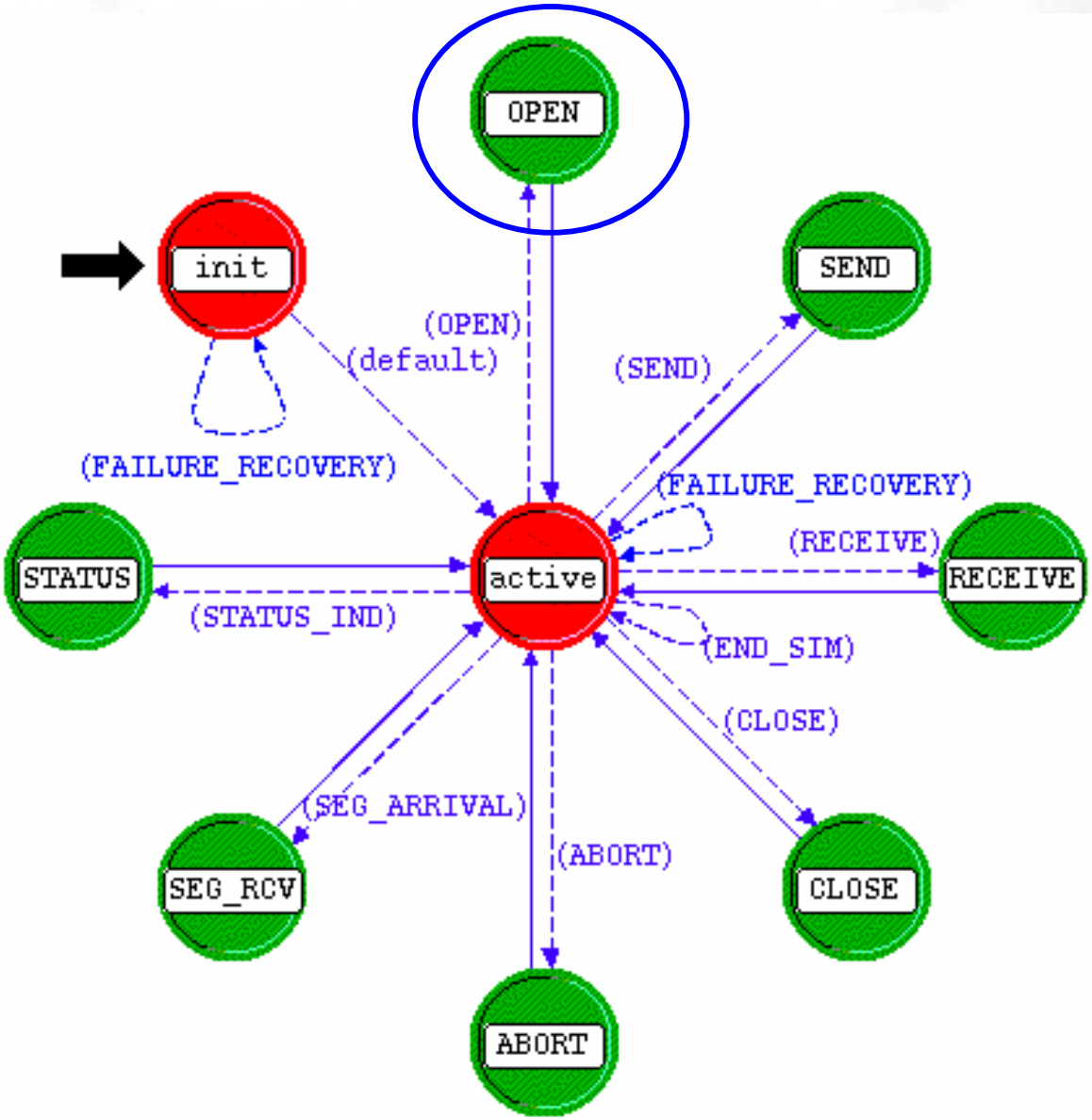
TCP-ADaLR: OPNET implementation

- OPNET TCP process models implement all standard TCP features and includes additional features
- Modification to the OPNET node and process models of the TCP sender:
  - Ethernet server advanced node model
  - *tcp_manager_v3* parent process communicates with the session and IP layers
  - *tcp_conn_v3* child process is invoked by the *tcp_manager_v3* process when a new TCP connection is established by the network node

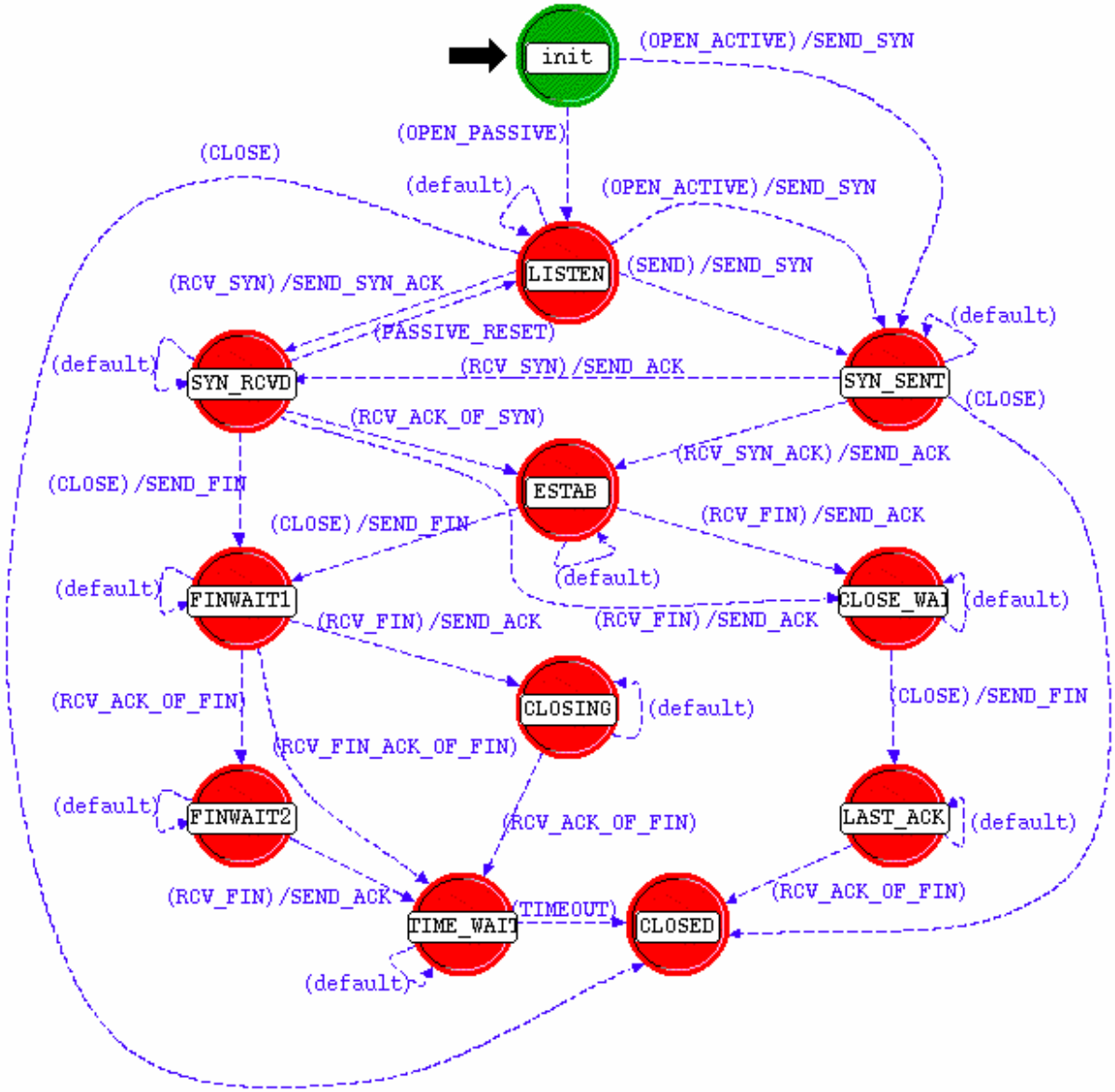
OPNET *Ethernet server advanced* node model



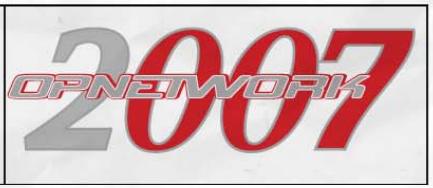
tcp_manager_v3 parent process



tcp_conn_v3 child process

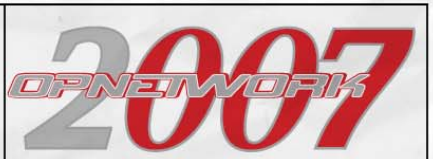


tcp_conn_v3 child process: modified functions



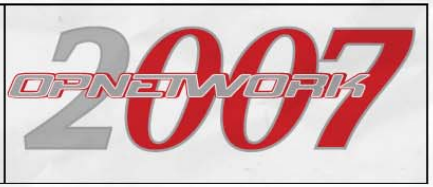
- *tcp_rtt_measurements_update* ():
  - used to compute RTT and RTO
  - modified to implement the computation of the scaling component ρ
- *tcp_cwnd_update* ():
  - used to increment the *cwnd* during slow start, congestion avoidance, fast retransmit, and fast recovery
  - modified to implement the adaptive *cwnd* increase mechanism and loss recovery mechanism during the fast recovery phase

tcp_conn_v3 child process: modified functions



- *tcp_snd_total_data_size* ():
  - used to compute the number of data segments to be sent after each ACK is received or when data is to be retransmitted
  - modified to implement the adaptive *rwnd* increase mechanism
- *tcp_snd_data_size* ():
  - used to compute the size of each data segment to be sent after an ACK is received or when data is to be retransmitted
  - modified to implement the adaptive *rwnd* increase mechanism

tcp_conn_v3 child process: modified functions

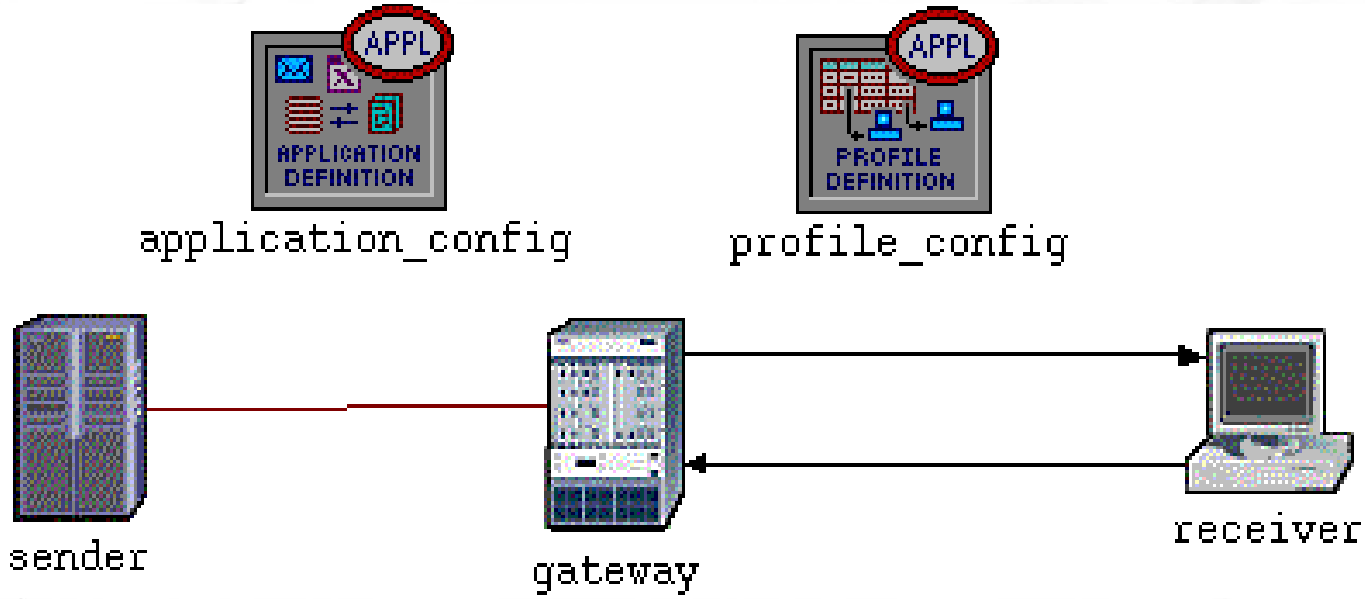


- *tcp_timeout_retrans* ():
  - used to retransmit segments after the RTO timer expires
  - modified to implement the loss recovery mechanism for computing subsequent RTO timer expirations
- *tcp_una_buf_process* ():
  - used determine the number of unacknowledged bytes from the retransmission buffer to send during fast retransmit or after RTO timer expiration
  - modified to implement the loss recovery mechanism for avoiding an unnecessarily large number of retransmissions

Roadmap

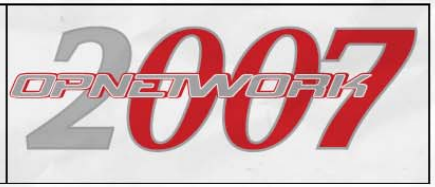
- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR):
  - algorithm description
  - OPNET implementation
- Performance evaluation of TCP-ADaLR:
  - simulation scenarios and results
  - fairness and friendliness scenarios
- Conclusions

OPNET network model



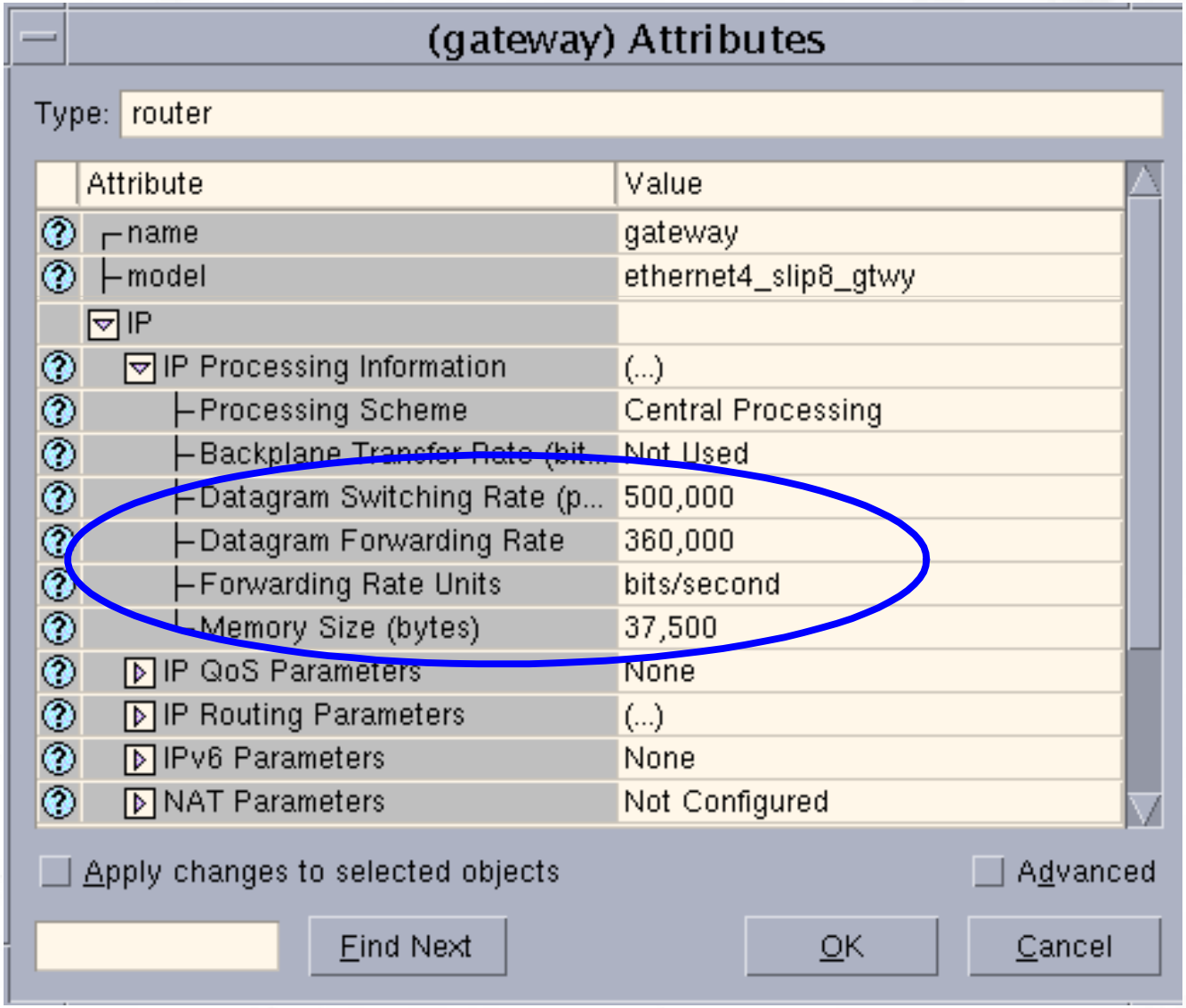
- Propagation delays are one-way
- Ethernet link between the gateway and the server is full-duplex with a data rate of 10 Mb/s
- GEO satellite link between the gateway and the client is asymmetric with data rates of 2 Mb/s downlink and 256 kb/s uplink

Simulation scenarios and performance metrics

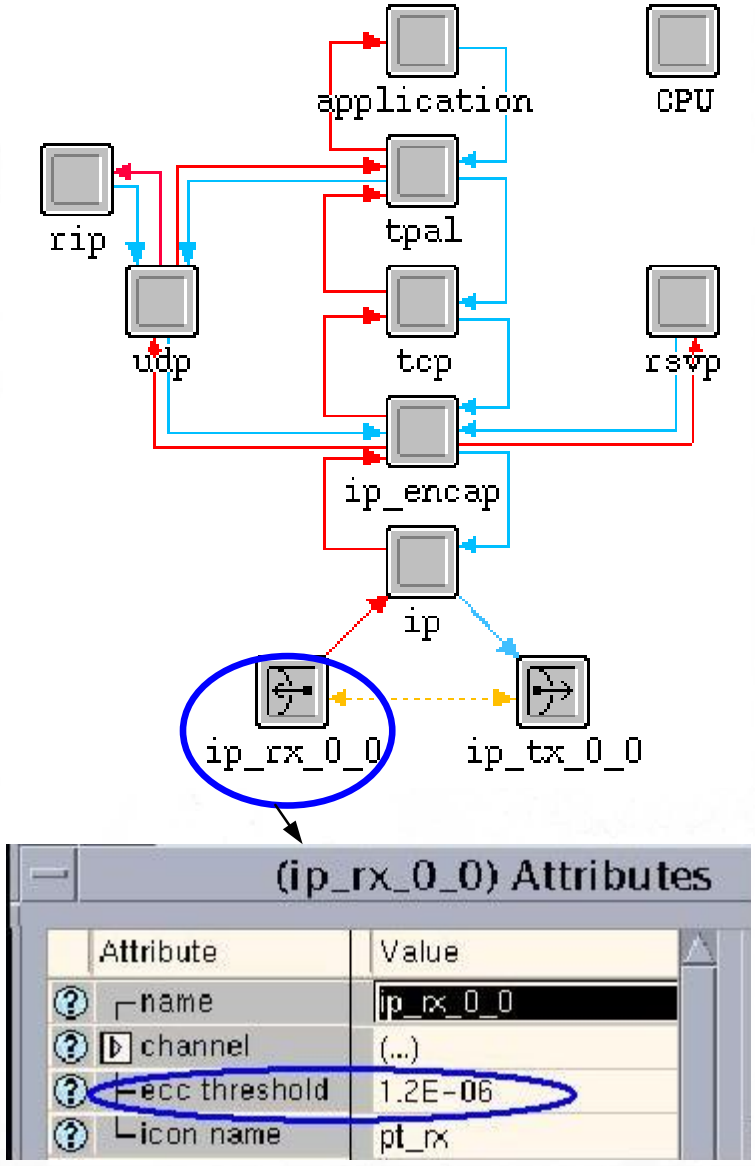


- Four scenarios with GEO satellite link:
  - ideal with no losses
  - ideal with only congestion losses
  - with only error losses
  - with both congestion and error losses
- Performance metrics:
  - HTTP page response time
  - FTP download response time
  - TCP goodput and throughput
  - satellite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

OPNET *Ethernet4 slip8 gateway* node model attributes



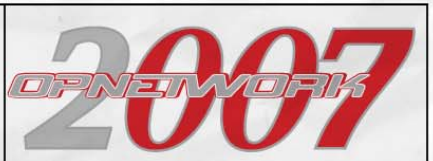
OPNET *PPP workstation advanced* node model: TCP receiver



- GEO satellite link was modeled as an AWGN channel with the OPNET *PPP link advanced* model
- PER = $1 - (1 - BER)^N$
- Error correction threshold:
  - the highest proportion of bit errors in a packet accepted by a TCP receiver
  - equivalent to the PER when the BER is 10^{-10}

AWGN: additive white Gaussian noise
PER: packet error rate
BER: bit error rate
N: number of bits in transmitted packet

Simulated variants and parameters



- TCP variants:
  - TCP-ADaLR SACK
  - TCP-ADaLR NewReno
  - TCP SACK
  - TCP NewReno
- Parameters:
  - HTTP and FTP applications with constant parameters
  - TCP parameters: standard OPNET TCP parameters with and without delayed ACK

Simulated application parameters

HTTP web 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	0.0s
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 (*rwnd*)	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 description
  - OPNET implementation
- Performance evaluation of TCP-ADaLR:
  - simulation scenario with no losses
  - fairness and friendliness scenarios
- Conclusions

Scenario with no losses: HTTP page response time

TCP variant	Page response time (s)	
	Without delayed ACK	With delayed ACK
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

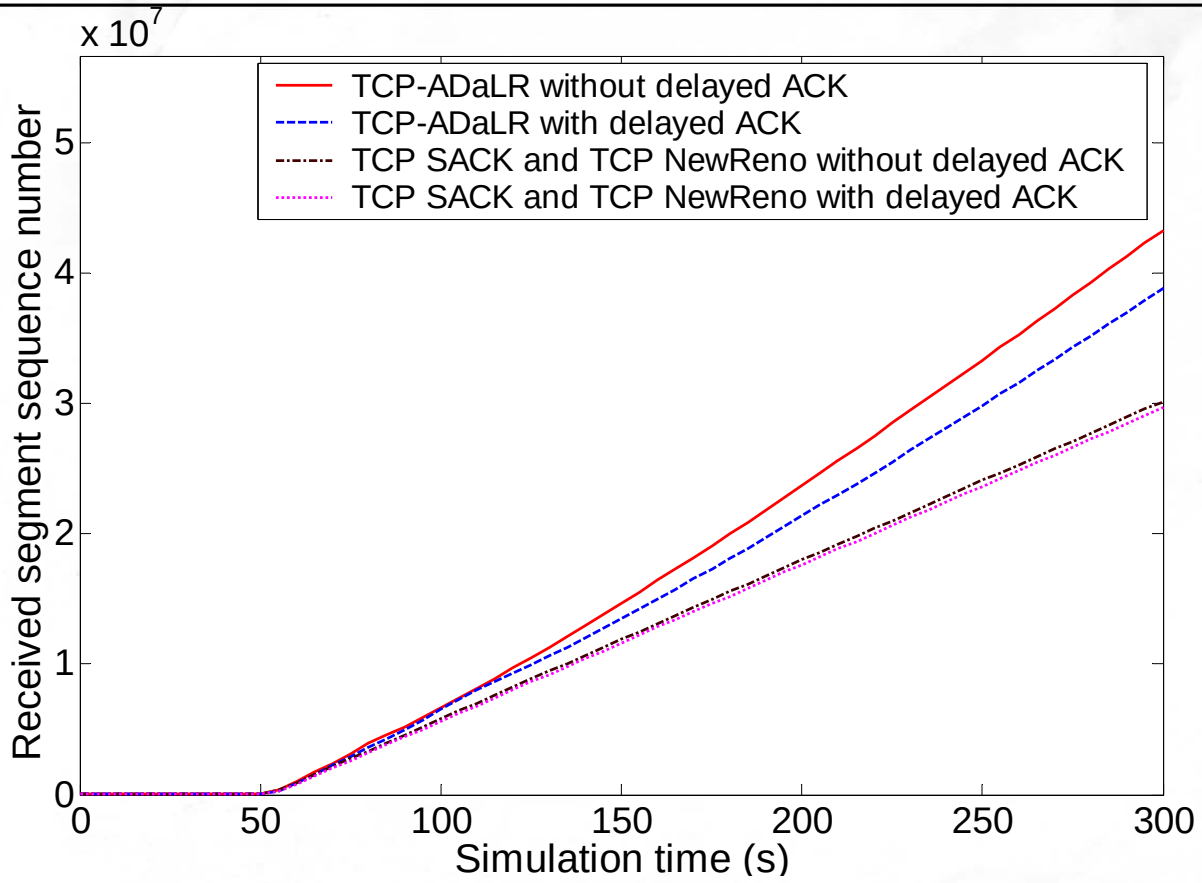
- TCP-ADaLR variants show shorter page response time:
  - 10% without delayed ACK
  - 9% with delayed ACK
- Adaptive window increase mechanisms allow transmission of additional segments

Scenario with no losses: FTP download response time

TCP variant	Download response time (s)	
	Without delayed ACK	With delayed ACK
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

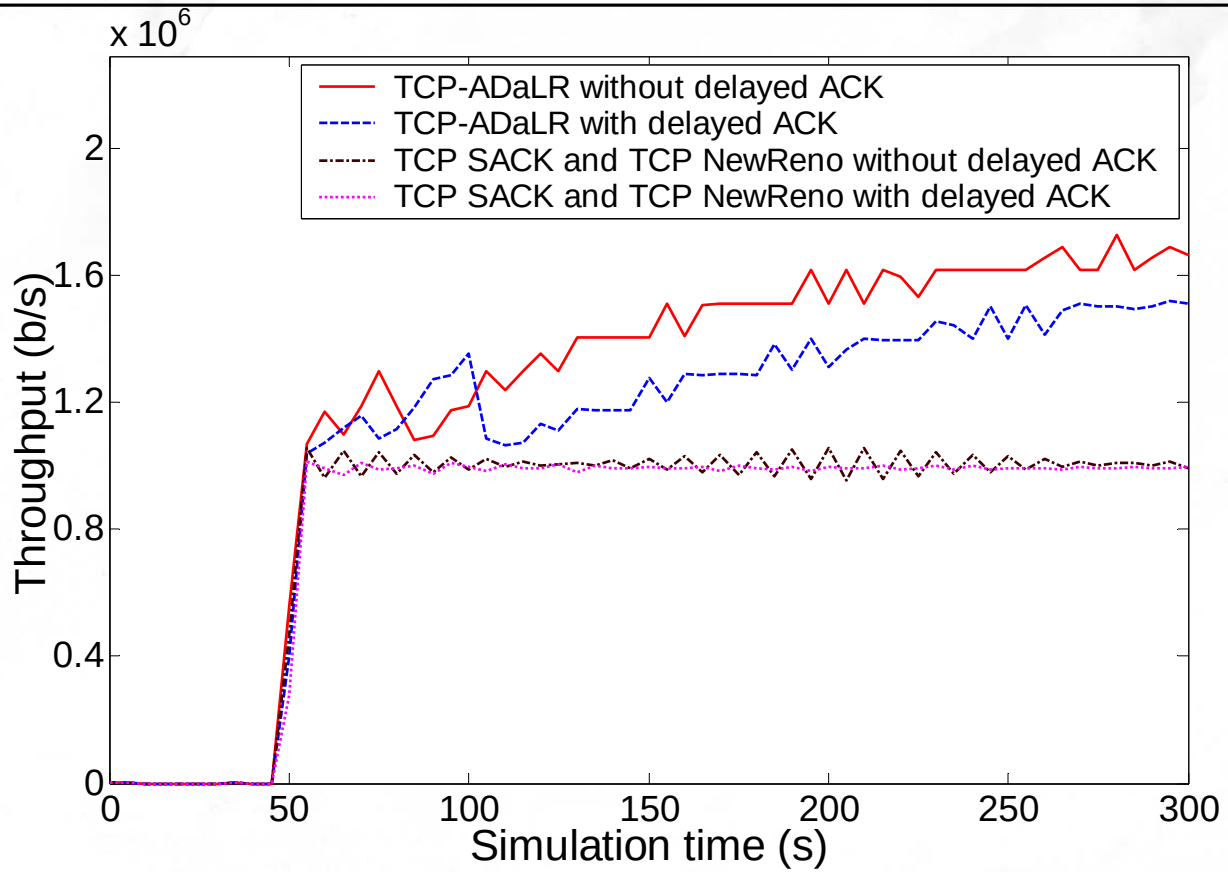
- TCP-ADaLR variants show shorter download response times:
  - 23% without delayed ACK
  - 28% with delayed ACK
- TCP-ADaLR algorithm does not degrade performance of TCP connections without delayed ACK

Scenario with no losses: TCP goodput



- TCP-ADaLR variants show higher TCP goodput than TCP SACK and TCP NewReno:
  - 50% without delayed ACK
  - 49% with delayed ACK

Scenario with no losses: satellite link throughput



- TCP-ADaLR variants show higher satellite link throughput than TCP SACK and TCP NewReno:
  - 66% without delayed ACK
  - 53% with delayed ACK

Roadmap

- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR):
  - algorithm description
  - OPNET implementation
- 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)	
	Without delayed ACK	With delayed ACK
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

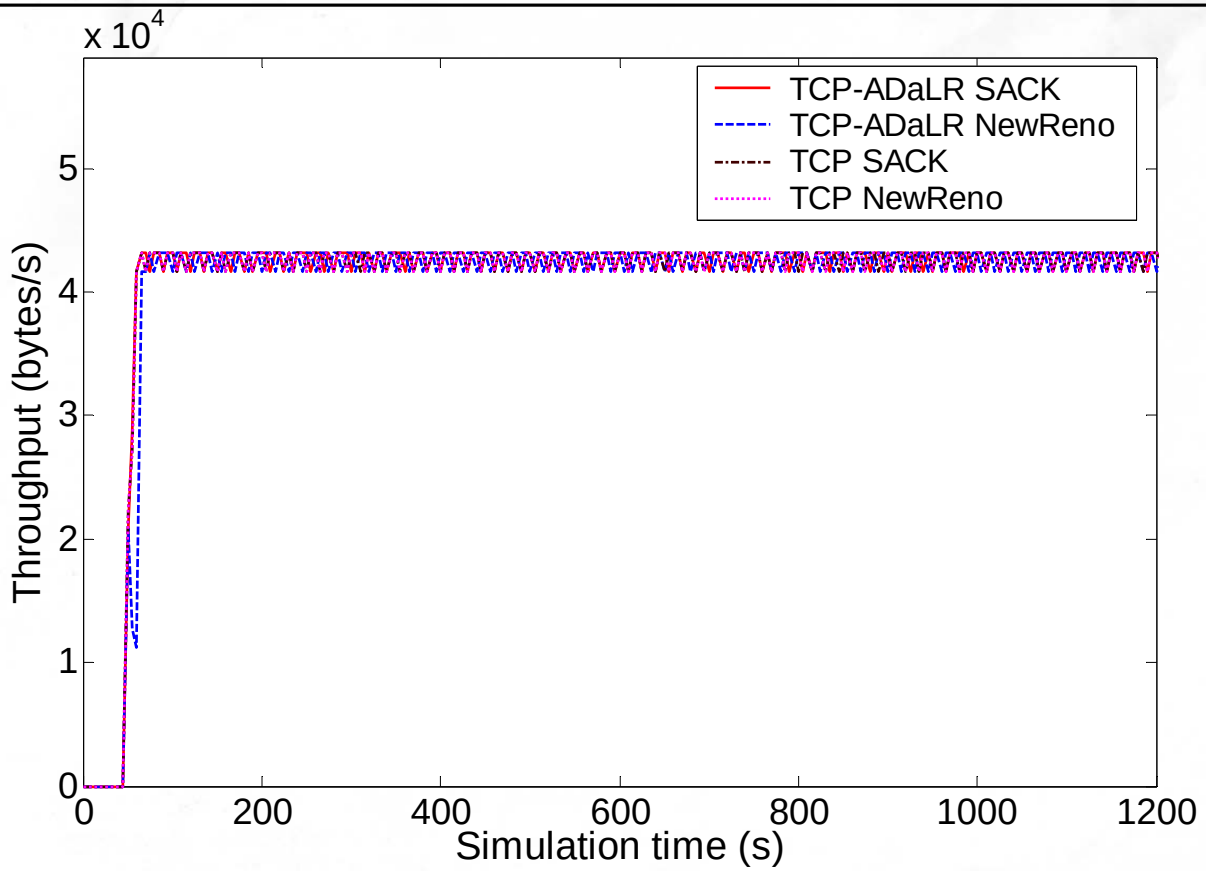
- TCP-ADaLR SACK exhibits shorter page response time than TCP SACK:
  - 12% without delayed ACK
  - 33% with delayed ACK
- Loss recovery mechanism enables quicker recovery from losses than TCP SACK and TCP NewReno

Scenario with only congestion losses: FTP download response time

TCP variant	Download response time (s)	
	Without delayed ACK	With delayed ACK
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

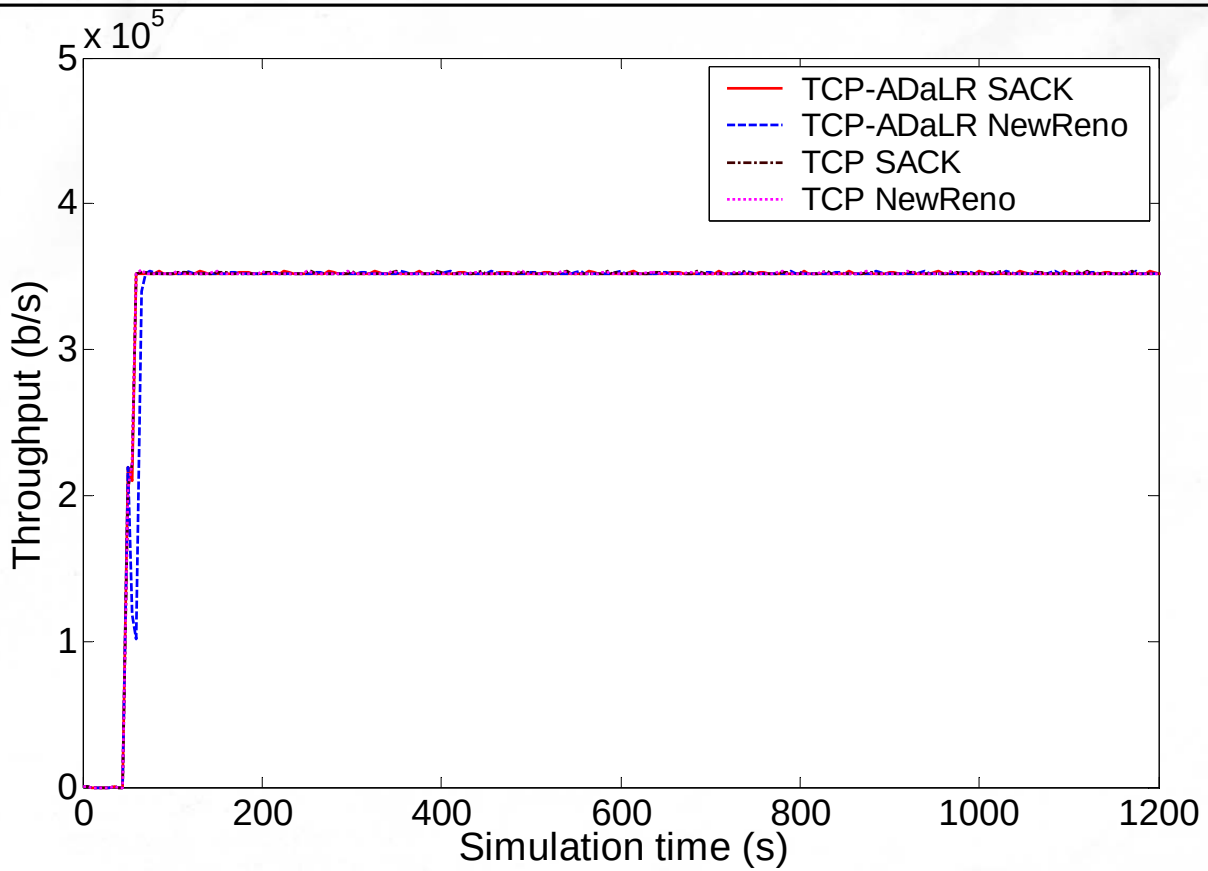
- TCP-ADaLR variants show download response times comparable to TCP SACK and TCP NewReno for both cases with and without delayed ACK
- The four TCP variants in the case without delayed ACK exhibit comparable performance as the four TCP variants in the case with delayed ACK

Scenario with only congestion losses: TCP throughput



- Case without delayed ACK:
  - TCP-ADaLR variants exhibit TCP throughput comparable to TCP SACK and TCP NewReno
  - performance degradation of the four TCP variants reflects the impact of congestion

Scenario with only congestion losses: satellite link throughput

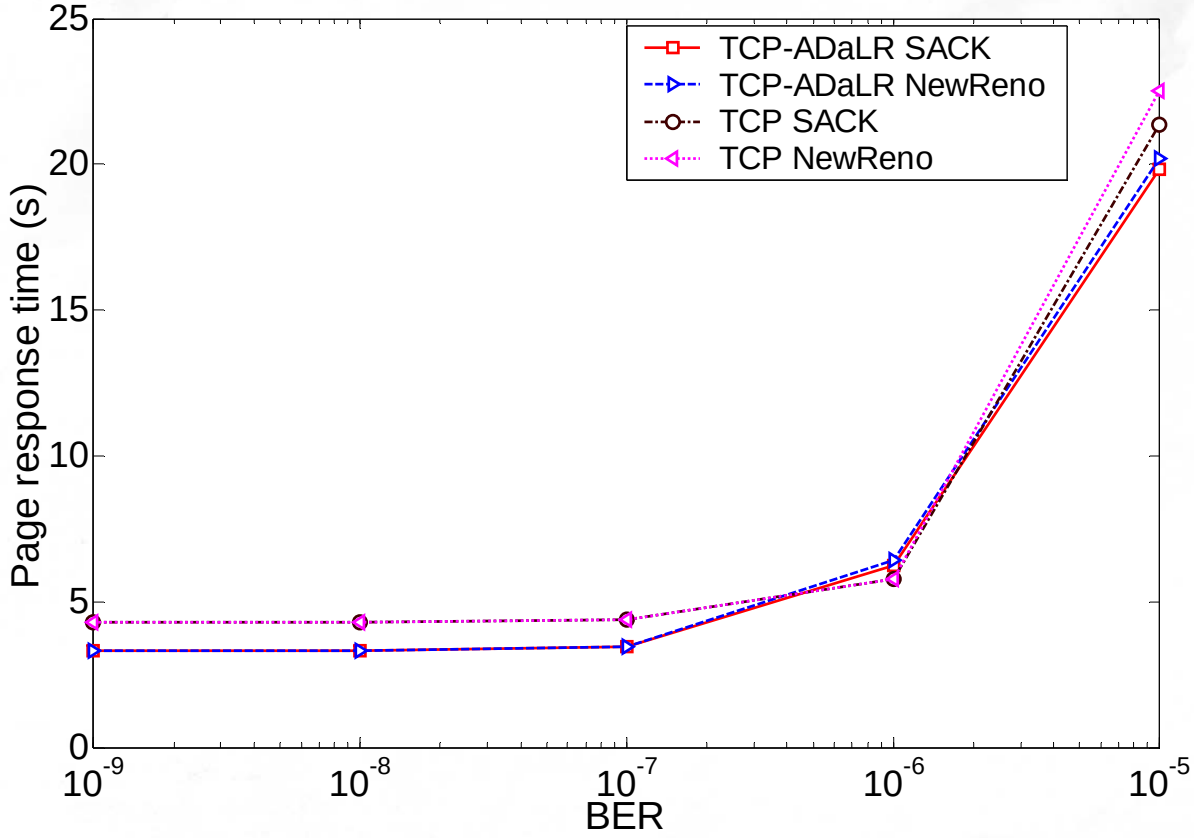


- Case without delayed ACK:
  - TCP-ADaLR variants exhibit satellite link throughput comparable to TCP SACK and TCP NewReno
  - satellite link throughput drops when congestion losses are detected

Roadmap

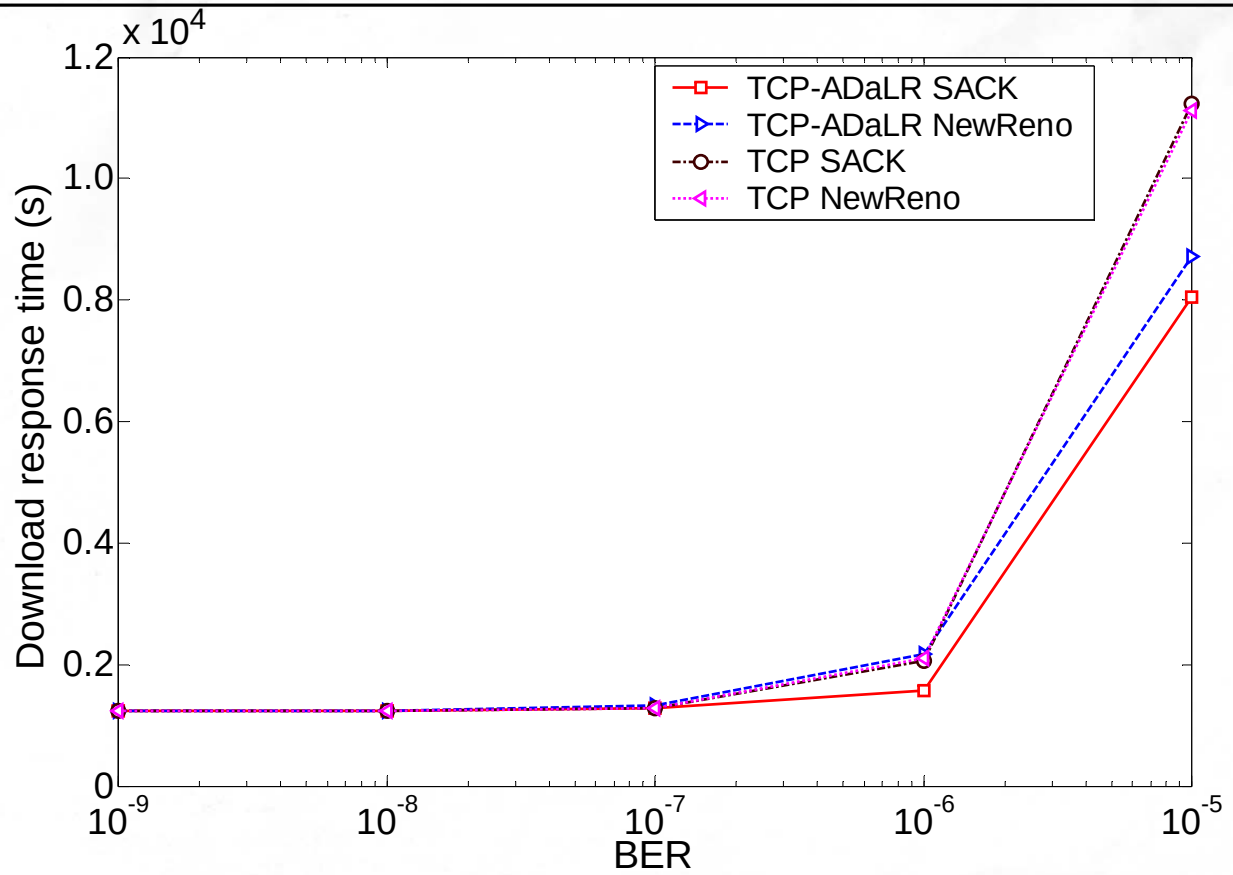
- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR):
  - algorithm description
  - OPNET implementation
- Performance evaluation of TCP-ADaLR:
  - simulation scenario with only error losses
  - fairness and friendliness scenarios
- Conclusions

Scenario with only error losses: HTTP page response time



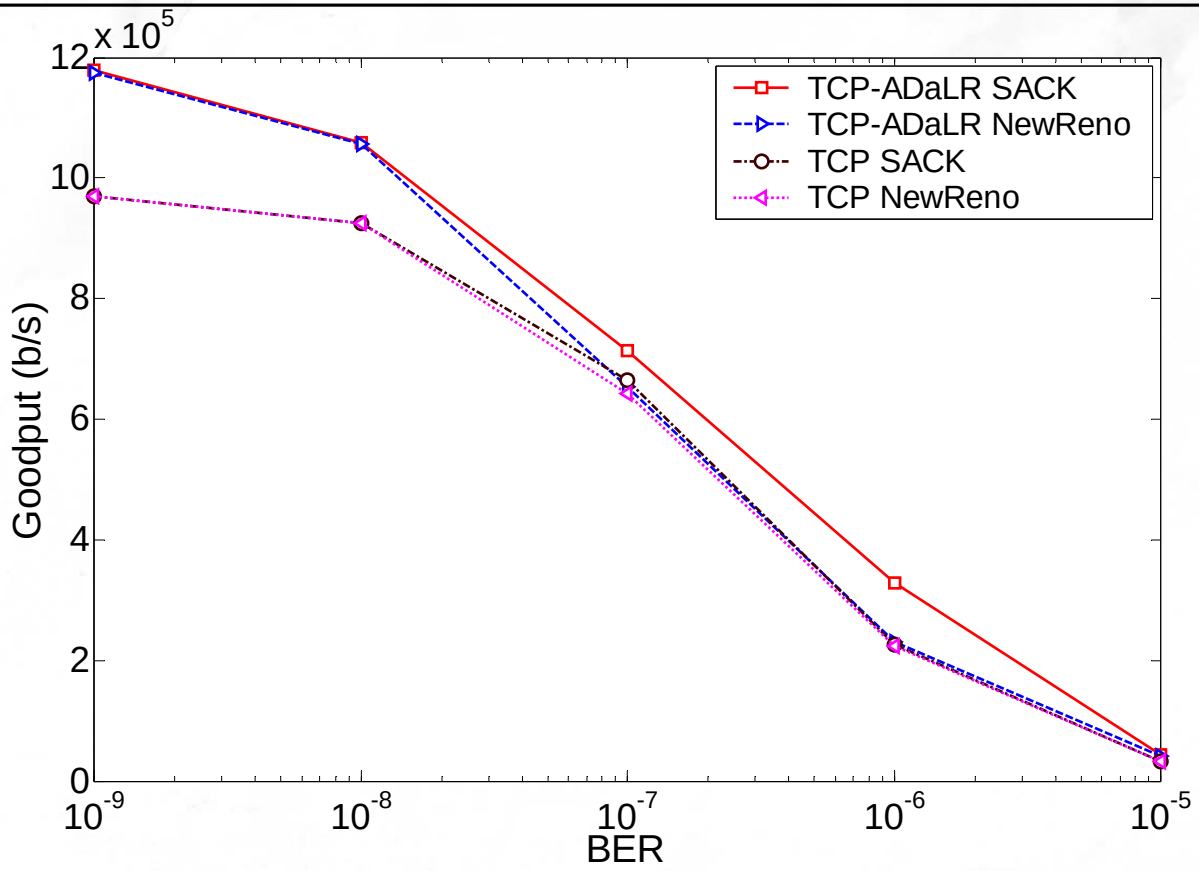
- Case without delayed ACK:
  - TCP-ADaLR SACK exhibits 7%–23% shorter page response time than TCP SACK
  - TCP-ADaLR NewReno exhibits 10%–23% shorter page response time than TCP NewReno

Scenario with only error losses: FTP download response time



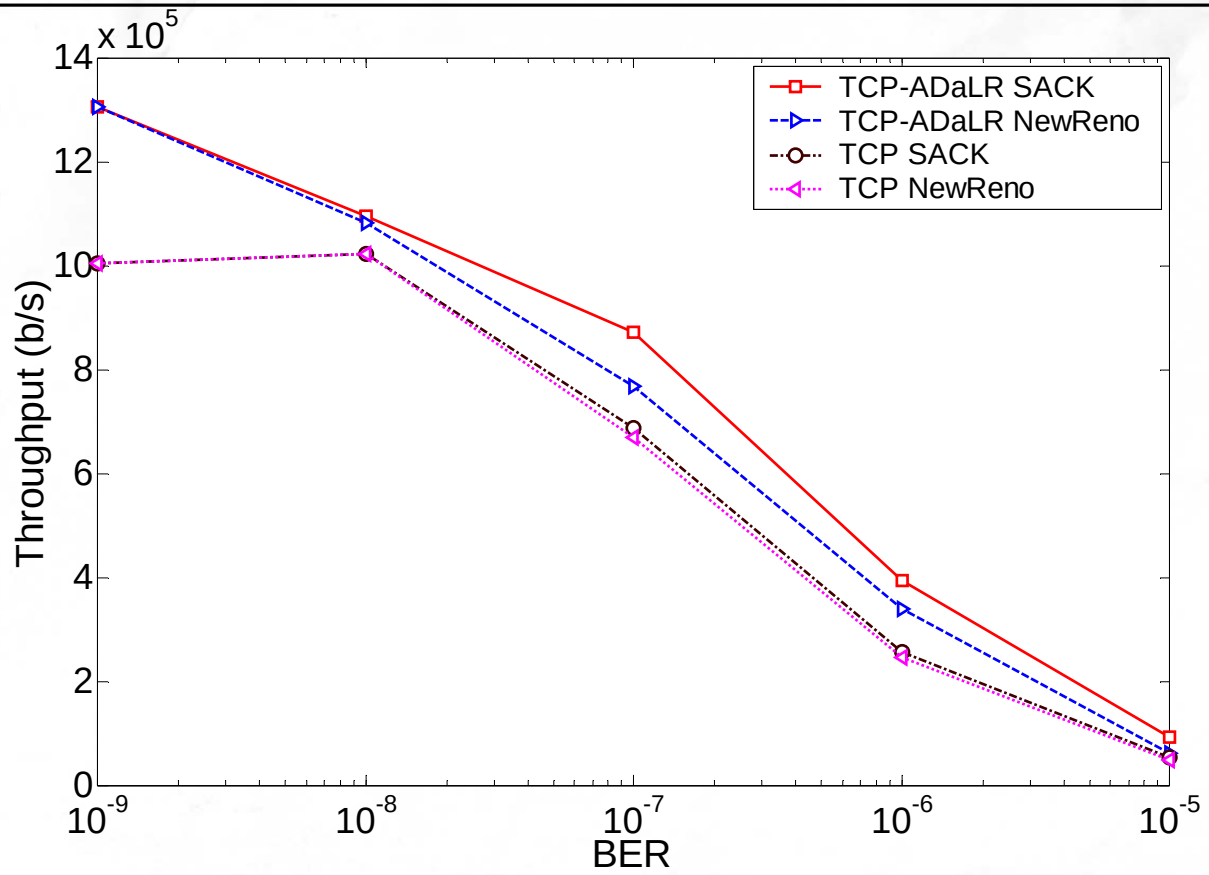
- Case without delayed ACK:
  - TCP-ADaLR SACK exhibits 6%–31% shorter download response time than TCP SACK
  - TCP-ADaLR NewReno exhibits 2%–26% shorter download response times than TCP NewReno

Scenario with only error losses: TCP goodput



- Case without delayed ACK:
  - TCP-ADaLR SACK exhibits 7%–46% higher TCP goodput than TCP SACK
  - TCP-ADaLR NewReno exhibits 2%–35% higher TCP goodput than TCP NewReno

Scenario with error losses only: satellite link throughput

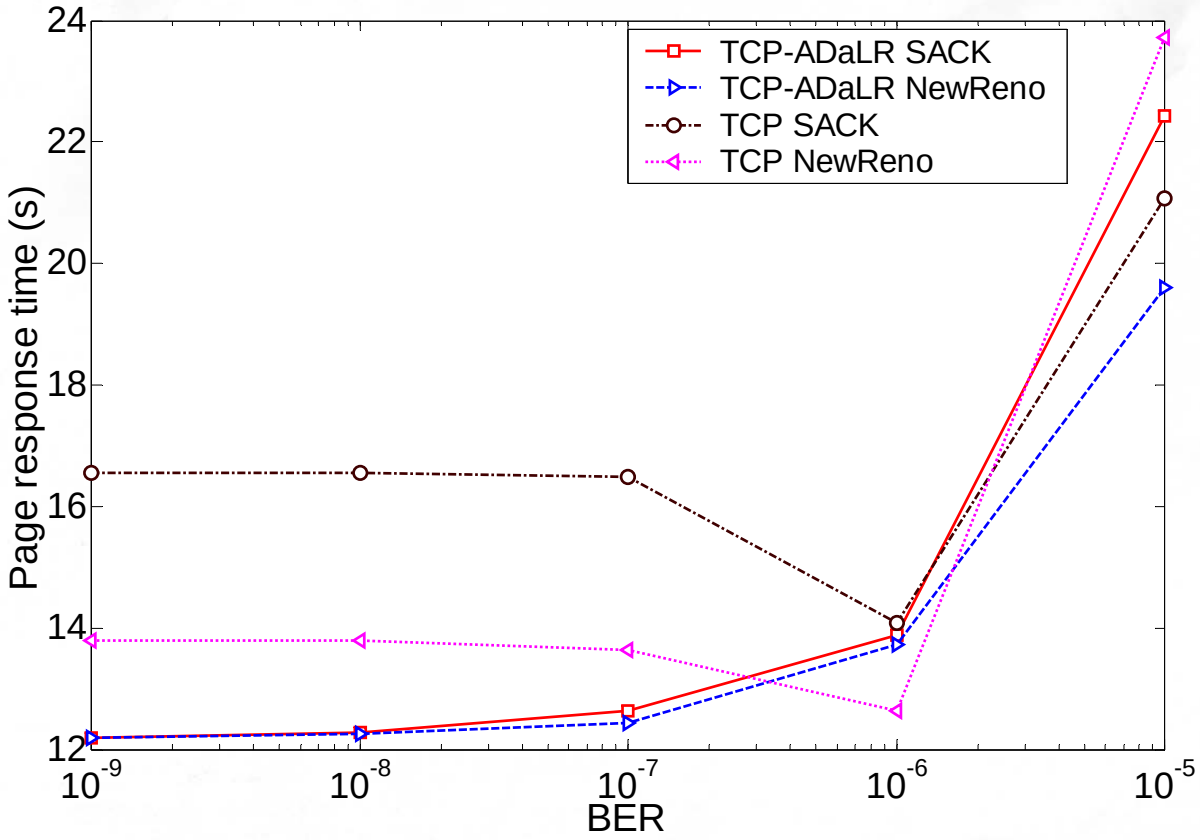


- TCP-ADaLR adaptive *cwnd* increase mechanism causes additional segments to be rapidly sent after losses have occurred
- TCP-ADaLR SACK and TCP-ADaLR NewReno recover more quickly than TCP SACK and TCP NewReno

Roadmap

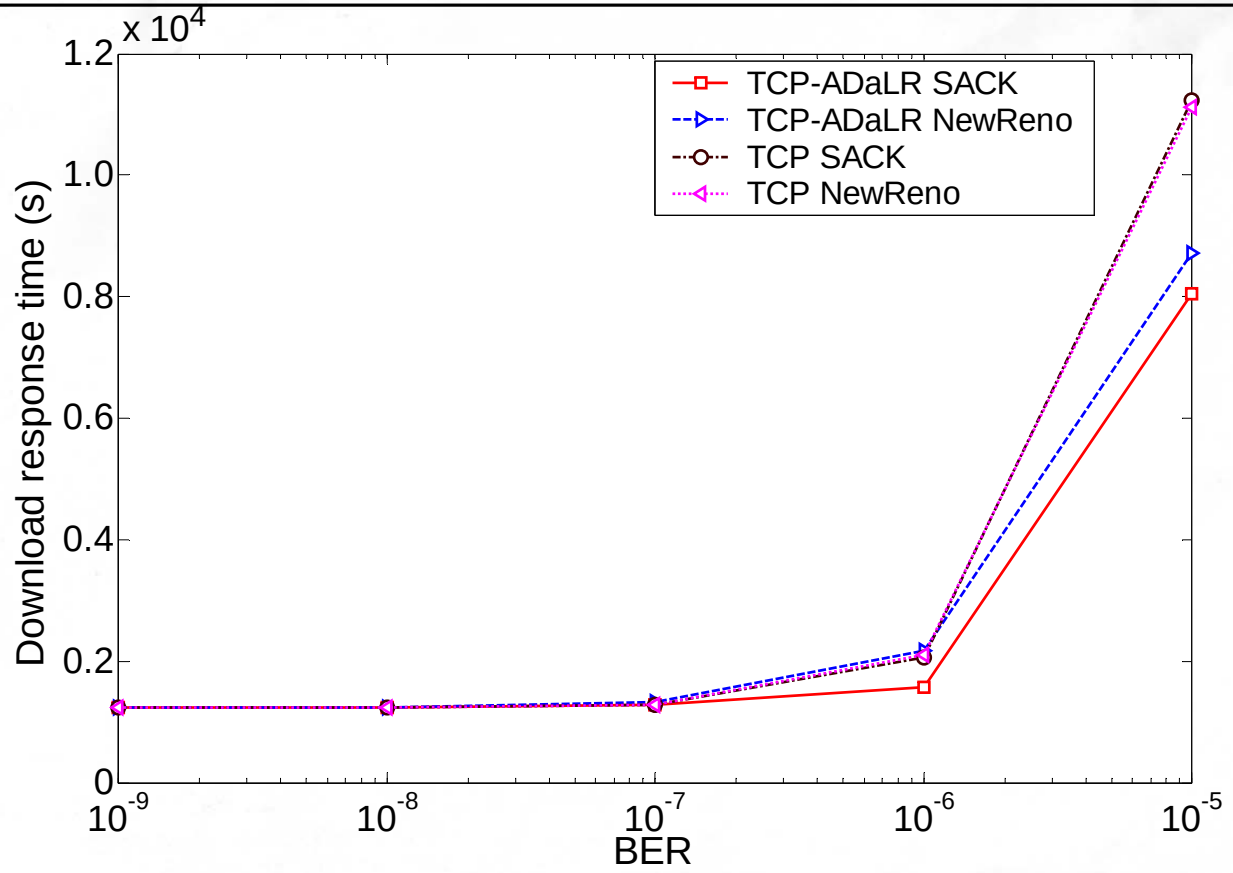
- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR):
  - algorithm description
  - OPNET implementation
- Performance evaluation of TCP-ADaLR:
  - simulation scenario with both congestion and error losses
  - fairness and friendliness scenarios
- Conclusions

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



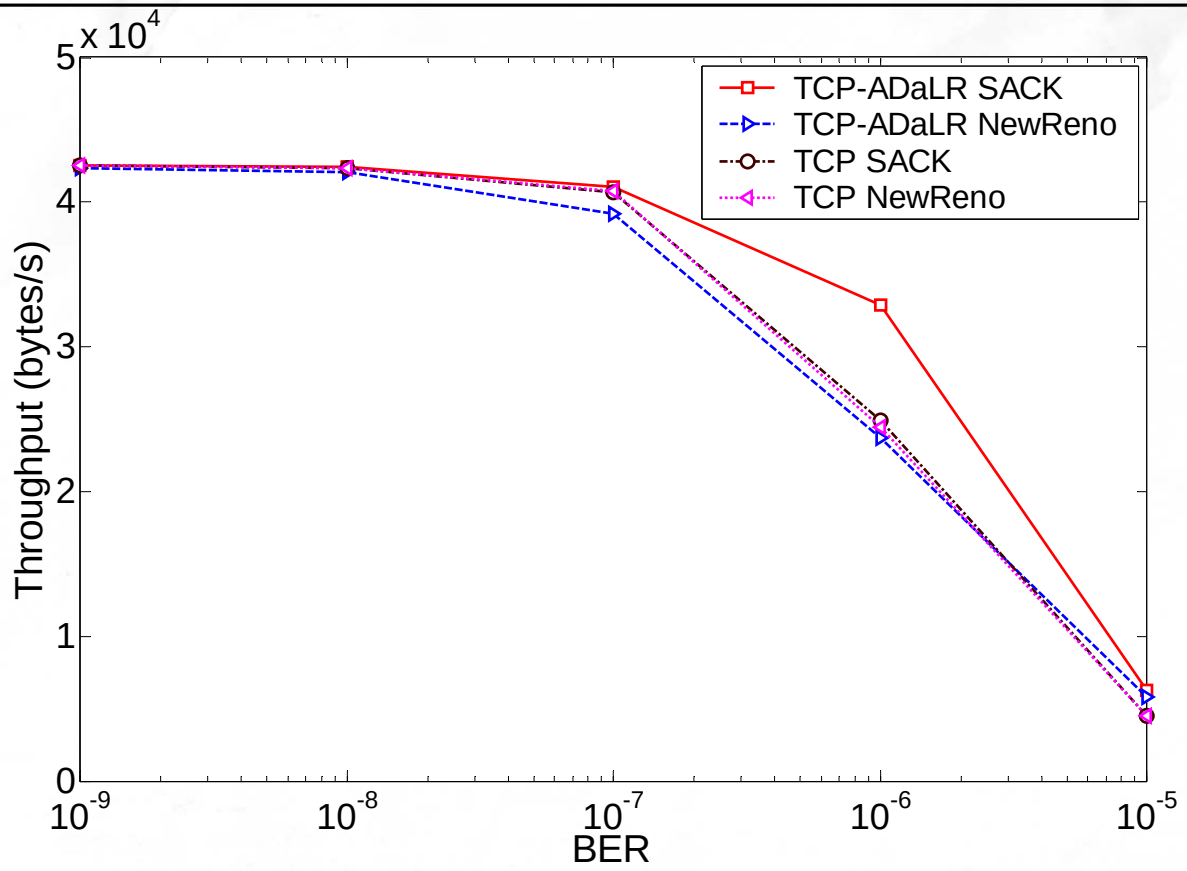
- Case without delayed ACK:
  - TCP-ADaLR SACK exhibits up to 26% shorter page response time than TCP SACK
  - TCP-ADaLR NewReno exhibits up to 17% shorter page response time than TCP NewReno

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



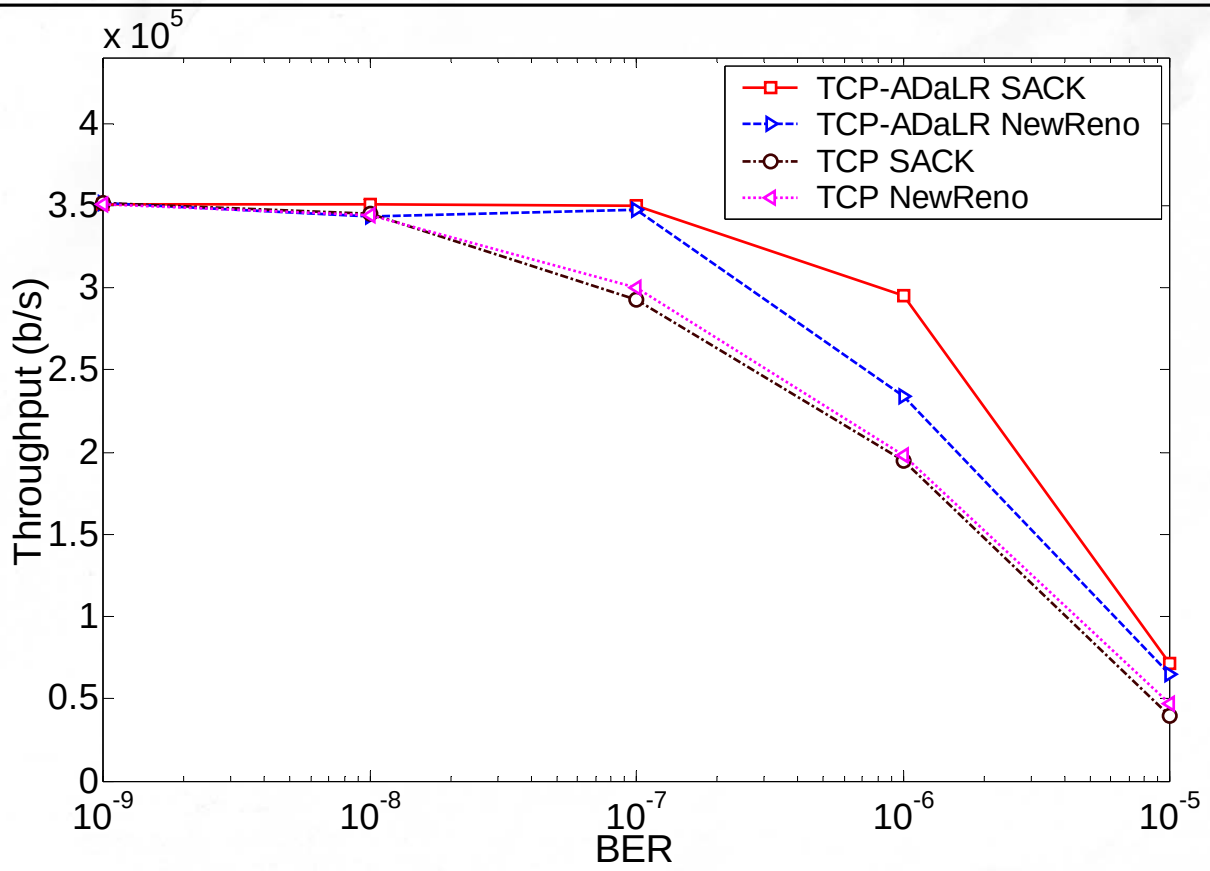
- Case without delayed ACK:
  - TCP-ADaLR SACK exhibits up to 28% lower download response times than TCP SACK
  - TCP-ADaLR NewReno exhibits up to 21% lower download response times than TCP NewReno

Scenario with both congestion and error losses: TCP throughput



- Case without delayed ACK:
  - TCP-ADaLR SACK exhibits up to 39% higher TCP throughput than TCP SACK
  - TCP-ADaLR NewReno exhibits up to 27% higher TCP throughput than TCP NewReno

Scenario with both congestion and error losses: satellite link throughput

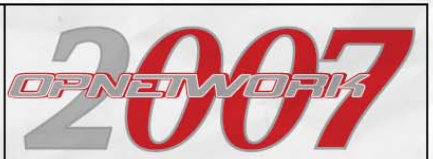


- Case without delayed ACK:
  - TCP-ADaLR SACK exhibits up to 79% higher satellite link throughput than TCP SACK
  - TCP-ADaLR NewReno exhibits up to 39% higher satellite link throughput than TCP NewReno

Roadmap

- Introduction
- Motivation
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR):
  - algorithm description
  - OPNET implementation
- 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 TCP connections with distinct TCP variants achieve equal bandwidth allocation

Fairness and friendliness

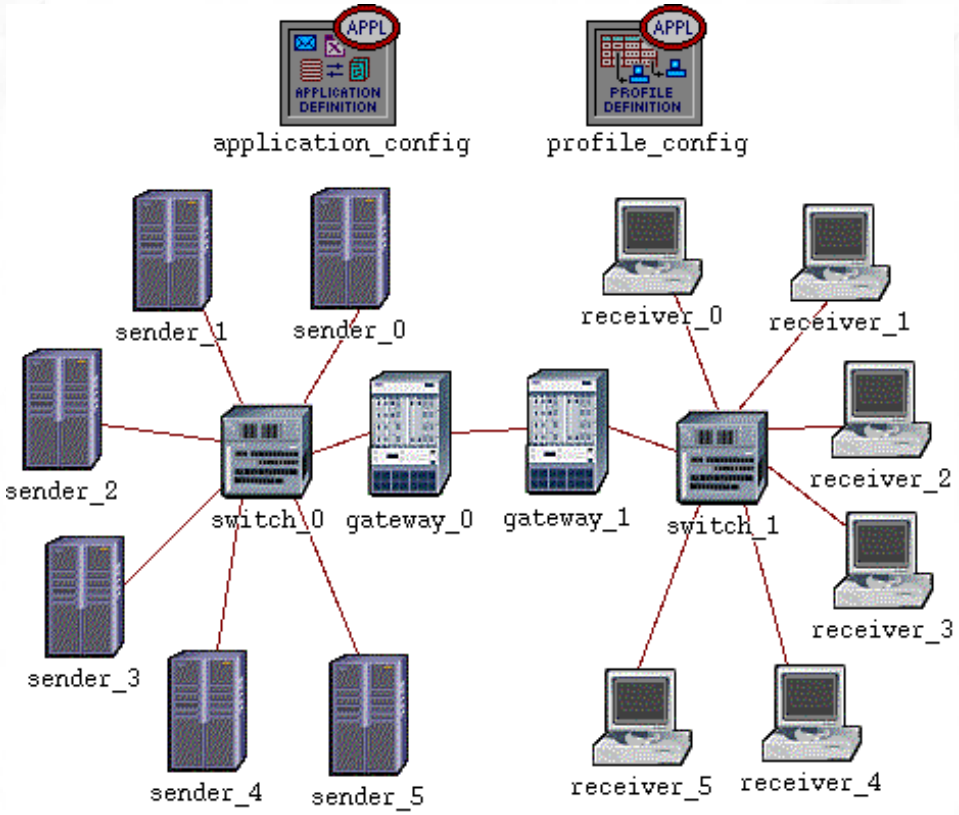
- Jain's metric of fairness:

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

  - *FI* is the fairness (friendliness) index
  - *n* is the number of competing connections
  - *tj* 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

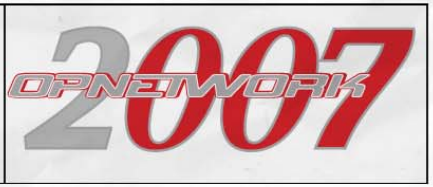
D. Chiu and R. Jain, "Analysis of the increase/decrease algorithms for congestion avoidance in computer networks," *J. of Comput. Netw. and ISDN Syst.*, vol. 17, no. 1, pp. 1-14, June 1989.

Fairness and friendliness: OPNET network model



- Links are 10 Mb/s full-duplex Ethernet
- One-way link propagation delays between receivers (0 to 5) and Ethernet switch_1 are 250 ms, 200 ms, 150 ms, 50 ms, 25 ms, and 12.5 ms
- Bottleneck link is located between gateway_0 and gateway_1

Fairness and friendliness scenarios



- Fairness scenarios with six coexisting TCP connections with distinct RTTs:
  - 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 TCP connections:
  - 3 TCP-ADaLR NewReno longer RTT connections: 300 ms, 400 ms, and 500 ms
  - 3 TCP NewReno shorter RTT connections: 25 ms, 50 ms, and 100 ms

Fairness (friendliness) indices

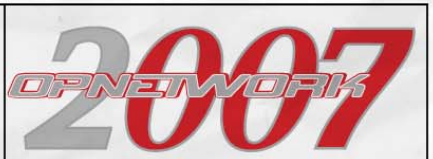
TCP variant	Fairness index
TCP-ADaLR NewReno	0.9510
TCP NewReno	0.8650

- 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	0.9859

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

Roadmap



- Introduction
- Background and related work
- TCP with adaptive delay and loss response (TCP-ADaLR):
  - algorithm description
  - OPNET implementation
- 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 only 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 only congestion losses
- 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

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. Technol.*, Cancun, Mexico, Jan. 2004, pp. 1–6.
- C. Fraleigh et al., "Packet-level traffic measurements from the Sprint IP backbone," *IEEE Netw.*, 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.
- 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.
- 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.

References

- J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby, "Performance enhancing proxies intended to mitigate link-related degradations," *RFC 3135*, June 2001.
- 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.
- 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.
- 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 .