



Stroboscopic model and bifurcations in TCP/RED

Mingjian Liu, Hui Zhang, and Ljiljana Trajković
{jliu1, hzhange, ljilja}@cs.sfu.ca

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



Roadmap

- Introduction
- Discrete-time model of TCP Reno with RED:
 - TCP/RED: model with **one** state variables
 - model validation
- Bifurcation diagrams
- Conclusion
- References



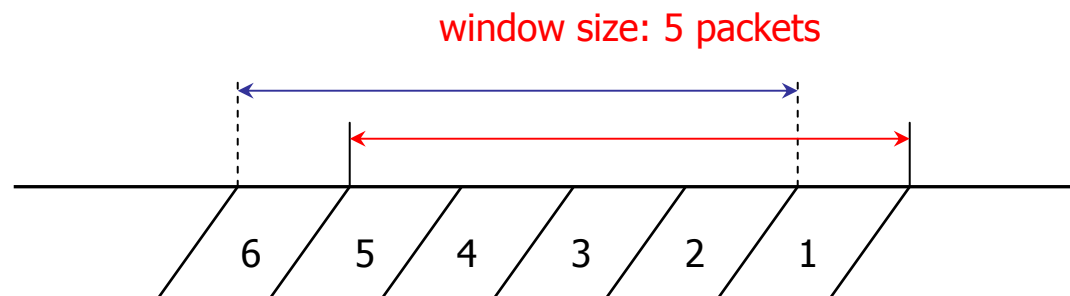
Motivation

- Modeling TCP Reno with RED:
 - examine the interactions between TCP and RED
 - understand and predict the dynamical network behavior
 - analyze the impact of system parameters

RED: Random Early Detection Gateways for Congestion Avoidance

TCP

- TCP: Transmission Control Protocol
- Fourth layer of the OSI model
- Connection oriented, reliable, and byte-stream service
- Employs window based flow and congestion control algorithms



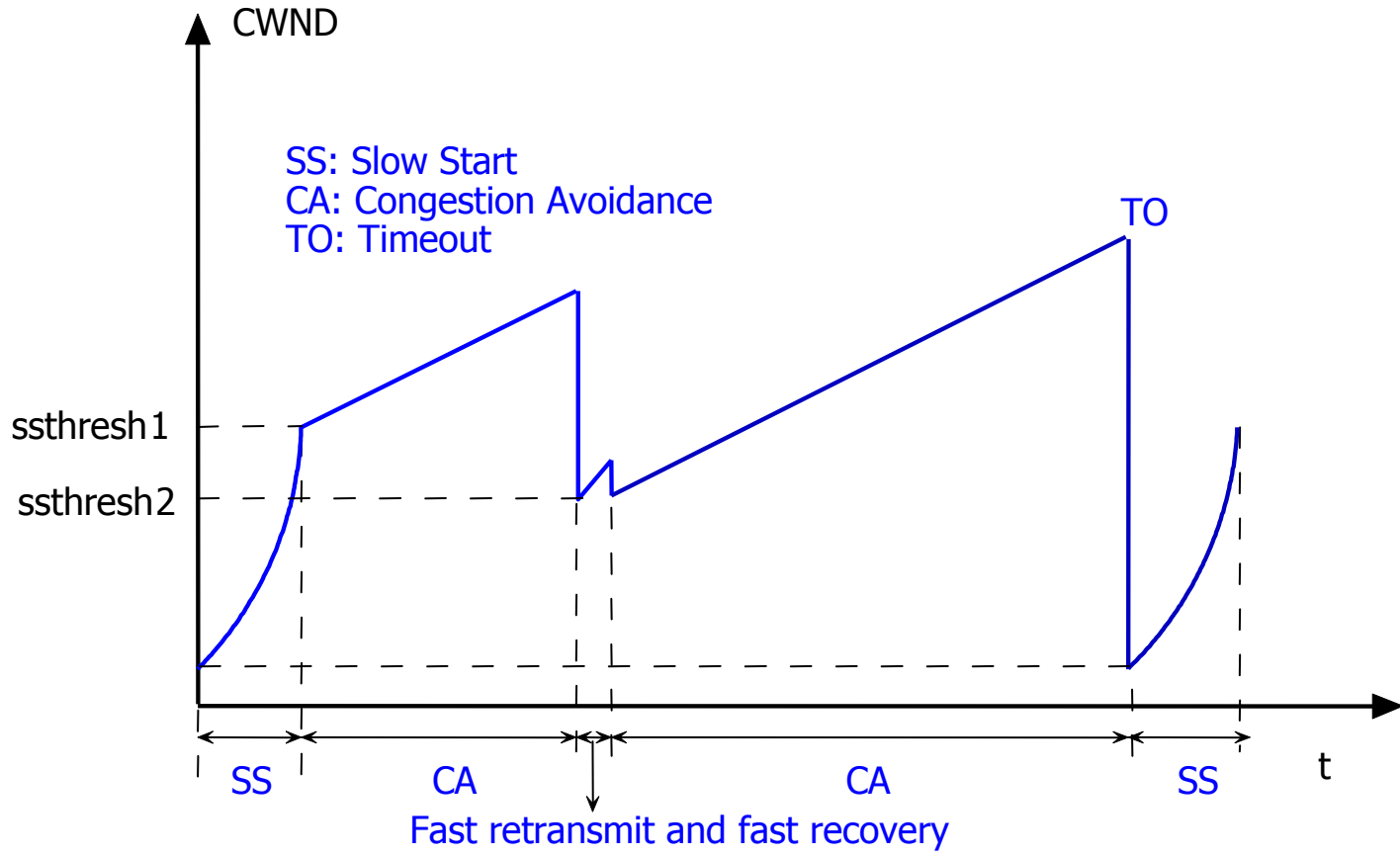
OSI: Open System Interconnection reference model



TCP

- Several flavors of TCP:
 - Tahoe: 4.3 BSD Tahoe (~ 1988)
 - slow start, congestion avoidance, and fast retransmit (RFC 793, RFC 2001)
 - Reno: 4.3 BSD Reno (~ 1990)
 - slow start, congestion avoidance, fast retransmit, and fast recovery (RFC 2001, RFC 2581)
 - NewReno (~ 1996)
 - new fast recovery algorithm (RFC 2582)
 - SACK (~ 1996, RFC 2018)

TCP Reno





TCP Reno: slow start and congestion avoidance

- Slow start:
 - $cwnd = IW$ (1 or 2 packets)
 - when $cwnd < ssthresh$
 $cwnd = cwnd + 1$ for each received *ACK*
- Congestion avoidance:
 - when $cwnd > ssthresh$
 $cwnd = cwnd + 1/cwnd$ for each *ACK*

cwnd : congestion window size

IW : initial window size

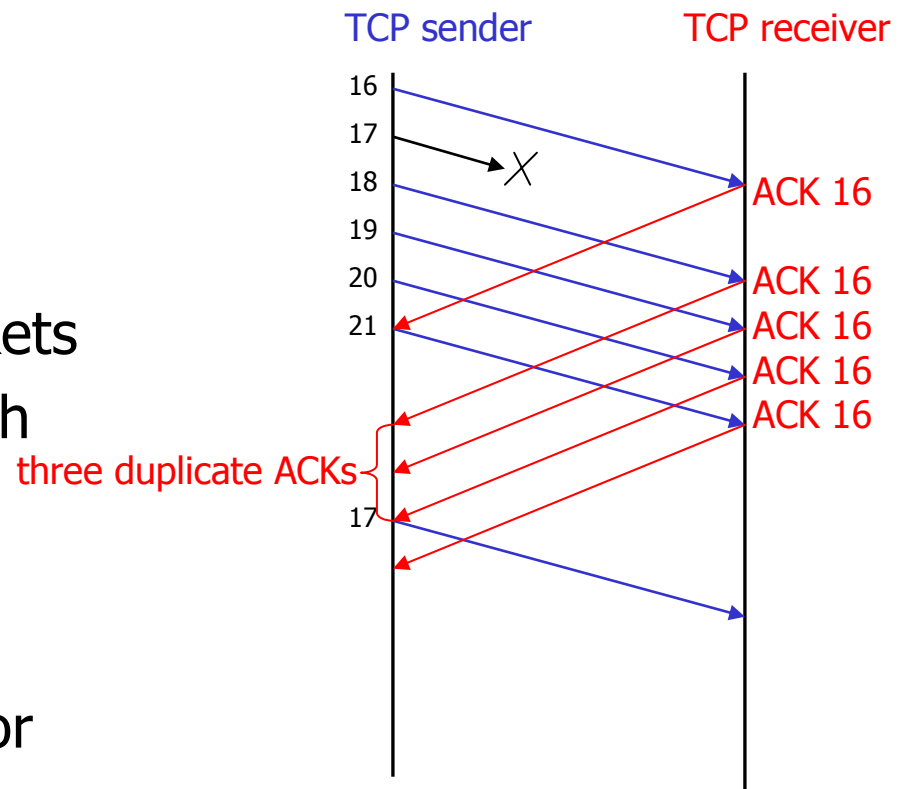
ssthresh : slow start threshold

ACK : acknowledgement

RTT : round trip time

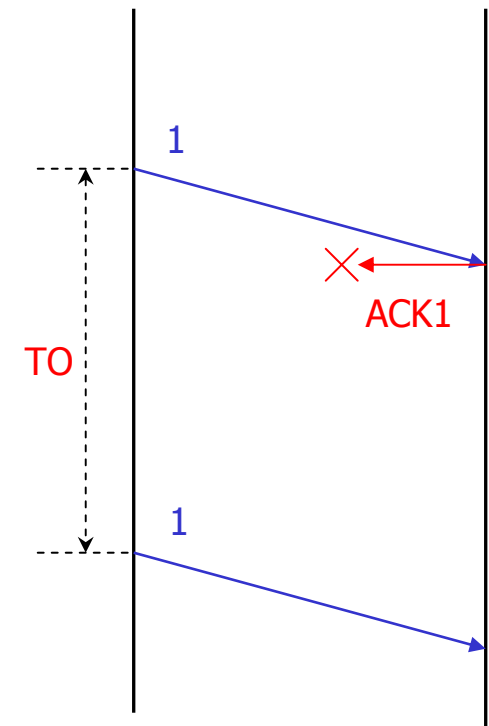
TCP Reno: fast retransmit and fast recovery

- three duplicate *ACKs* are received
- retransmit the packet
- $ssthresh = cwnd/2$,
 $cwnd = ssthresh + 3$ packets
- $cwnd = cwnd + 1$, for each additional duplicate ACK
- transmit the new data, if $cwnd$ allows
- $cwnd = ssthresh$, if ACK for new data is received



TCP Reno: timeout

- TCP maintains a **retransmission timer**
- The duration of the timer is called **retransmission timeout**
- Timeout occurs when the ACK for the delivered data is not received before the **retransmission timer** expires
- TCP sender retransmits the lost packet
- $ssthresh = cwnd/2$
 $cwnd = 1$ or 2 packets





AQM: Active Queue Management

- **AQM** (RFC 2309):
 - reduces bursty packet drops in routers
 - provides lower-delay interactive service
 - avoids the “lock-out” problem
 - reacts to the incipient congestion before buffers overflow
- AQM algorithms:
 - **RED** (RFC 2309)
 - **ARED**, **CHOKe**, **BLUE**, ...



RED

- Random Early Detection Gateways for Congestion Avoidance
 - Proposed by S. Floyd and V. Jacobson, LBN, 1993.
S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.
 - Main concept: drop packets **before** the queue becomes full



RED variables and parameters

- Main variables and parameters:
 - average queue size: \bar{q}
 - instantaneous queue size: q
 - drop probability: p_a
 - queue weight: w_q
 - maximum drop probability: p_{\max}
 - queue thresholds: q_{\min} and q_{\max}

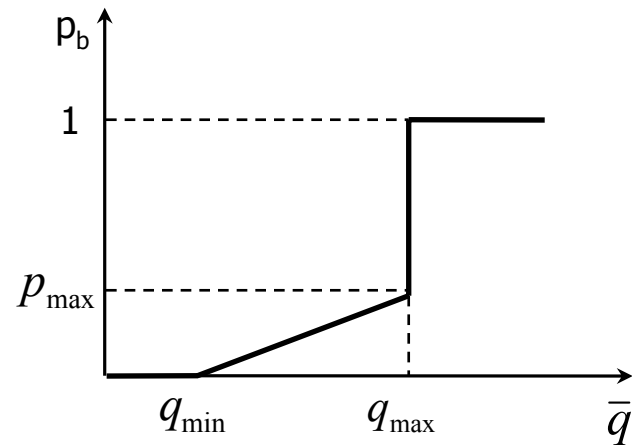
RED algorithm

Calculate:

- **average queue size** for each packet arrival

$$\bar{q} = (1 - w_q) \cdot \bar{q} + w_q \cdot q$$

- drop probability





RED algorithm: drop probability

- if $(q_{\min} < \bar{q} < q_{\max})$

$$p_b = p_{\max} \times \frac{\bar{q} - q_{\min}}{q_{\max} - q_{\min}} \quad p_a = \frac{p_b}{1 - count \times p_b}$$

count : number of packets that arrived since the last packet drop

- else if $(\bar{q} > q_{\max})$

$$p_a = 1$$

- else $(\bar{q} < q_{\min})$

$$p_a = 0$$

- Mark or drop the arriving packet with probability p_a



Modeling methodology

- Categories of TCP models:
 - averaged and **discrete-time** models
 - short-lived and **long-lived TCP** connections
- TCP/**RED** model:
 - **discrete-time** model with a **long-lived** connection
- State variables:
 - **window size** (TCP)
 - **average queue size** (RED)



TCP/RED model

- Key properties of the proposed TCP/RED model:
 - slow start, congestion avoidance, fast retransmit, and fast recovery (simplified)
 - Timeout:

J. Padhye, V. Firoiu, and D. F. Towsley, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Trans. Networking*, vol. 8, no. 2, pp. 133–145, Apr. 2000.
 - Captures the basic RED algorithm

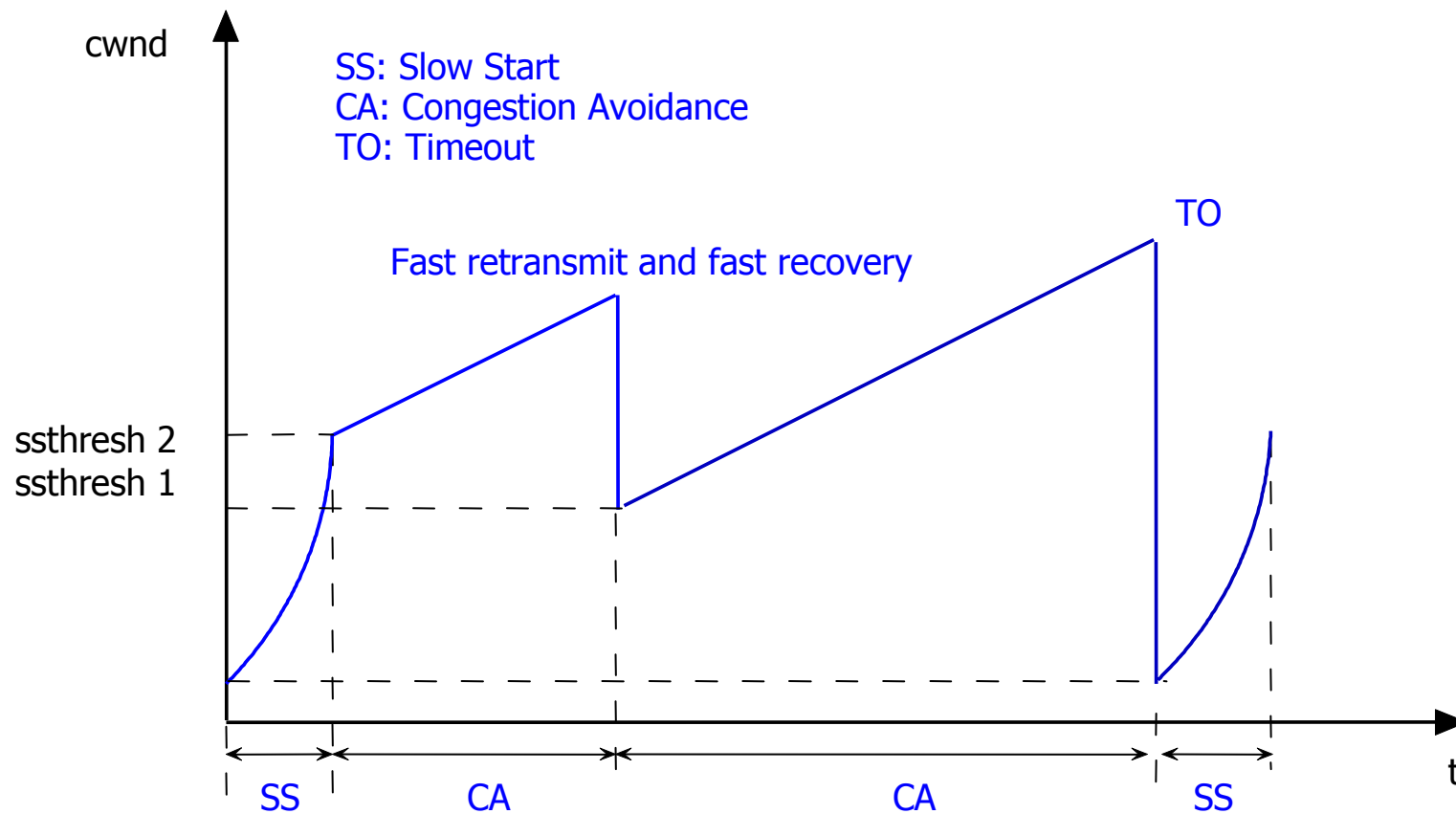


Assumptions

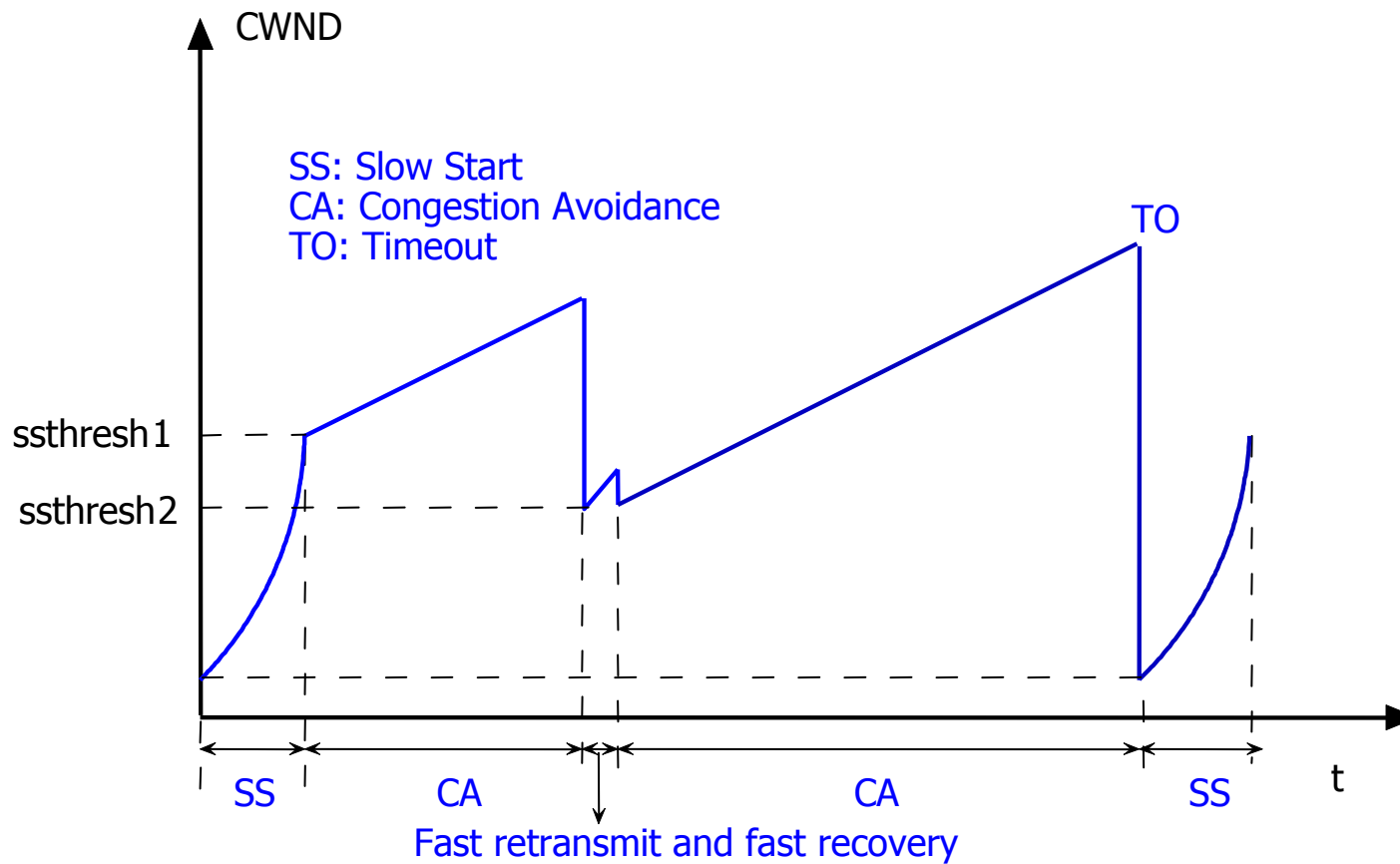
- Long-lived TCP connection
- Constant propagation delay between the source and the destination
- Constant packet size
- ACK packets are never lost
- Timeout occurs only due to packet loss
- The system is sampled at the end of every RTT interval

TCP/RED model simplifications

■ Simplified fast recovery



TCP Reno: fast recovery





TCP/RED model simplifications

- TO = 5 RTT

V. Firoiu and M. Borden, "A study of active queue management for congestion control," in *Proc. of IEEE INFOCOM 2000*, vol. 3, pp. 1435–1444, Tel-Aviv, Israel, Mar. 2000.

- RED: parameter **count** is not used

if $(q_{\min} < \bar{q} < q_{\max})$

$$p_b = p_{\max} \times \frac{\bar{q} - q_{\min}}{q_{\max} - q_{\min}}$$

$$\xrightarrow{p_a = p_b}$$

if $(q_{\min} < \bar{q} < q_{\max})$

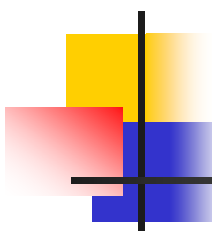
$$p_a = p_{\max} \times \frac{\bar{q} - q_{\min}}{q_{\max} - q_{\min}}$$

$$p_a = \frac{p_b}{1 - \text{count} \times p_b}$$



Proposed TCP/RED model

- **M-model**, a discrete nonlinear dynamical model of TCP Reno with **RED**:
 - P. Ranjan, E. H. Abed, and R. J. La, “Nonlinear instabilities in TCP-RED,” in *Proc. IEEE INFOCOM 2002*, New York, NY, USA, June 2002, vol. 1, pp. 249–258 and *IEEE/ACM Trans. on Networking*, vol. 12, no. 6, pp. 1079–1092, Dec. 2004.
- One state variable: **average queue size**
- The proposed **TCP/RED** model is:
 - simple and intuitively derived
 - able to capture detailed dynamical behavior of TCP/RED systems
 - has been verified via ns-2 simulations



TCP/RED model: state variable and parameters

- Variables:
 - \bar{q}_{k+1} : average queue size in round $k+1$
 - \bar{q}_k : average queue size in round k
 - w_q : queue weight in RED
 - N : number of TCP connections
 - K : constant = $\sqrt{3/2}$
 - p_k : drop probability in round k
 - C : capacity of the link between the two routers
 - d : round-trip propagation delay
 - M : packet size
 - $rwnd$: receiver's advertised window size



TCP/RED model: case 1

- Drop probability: $p_k \neq 0$

$$\begin{aligned}q_{k+1} &= q_k + B(p_k) \cdot RTT_{k+1} \cdot N - \frac{C \cdot RTT_{k+1}}{M} \\ &= q_k + \frac{K}{\sqrt{p_k} \cdot RTT_{k+1}} \cdot RTT_{k+1} \cdot N - \frac{C}{M} \left(d + \frac{q_k \cdot M}{C} \right) \\ &= \frac{K \cdot N}{\sqrt{p_k}} - \frac{C \cdot d}{M}\end{aligned}$$

where:

$B(p_k)$: TCP sending rate

$B(p_k) \cdot RTT_{k+1} \cdot N$: the number of incoming packets

$C \cdot \frac{RTT_{k+1}}{M}$: the number of outgoing packets



TCP/RED model: case 1

The average queue size is:

$$\bar{q}_{k+1} = (1 - w_q) \cdot \bar{q}_k + w_q \cdot q_{k+1}$$

hence

$$\bar{q}_{k+1} = (1 - w_q) \cdot \bar{q}_k + w_q \cdot \max\left(\frac{N \cdot K}{\sqrt{p_k}} - \frac{C \cdot d}{M}, 0\right)$$



Simplified S-TCP/RED model: case 2

- Drop probability: $p_k = 0$

$$\begin{aligned}q_{k+1} &= q_k + B(p_k) \cdot RTT_{k+1} \cdot N - \frac{C \cdot RTT_{k+1}}{M} \\ &= q_k + \frac{rwnd}{RTT_{k+1}} \cdot RTT_{k+1} \cdot N - \frac{C}{M} \left(d + \frac{q_k \cdot M}{C} \right) \\ &= rwnd \cdot N - \frac{C \cdot d}{M}\end{aligned}$$

The average queue size is:

$$\bar{q}_{k+1} = (1 - w_q) \cdot \bar{q}_k + w_q \cdot q_{k+1}$$

hence

$$\bar{q}_{k+1} = (1 - w_q) \cdot \bar{q}_k + w_q \cdot \left(rwnd \cdot N - \frac{C \cdot d}{M} \right)$$

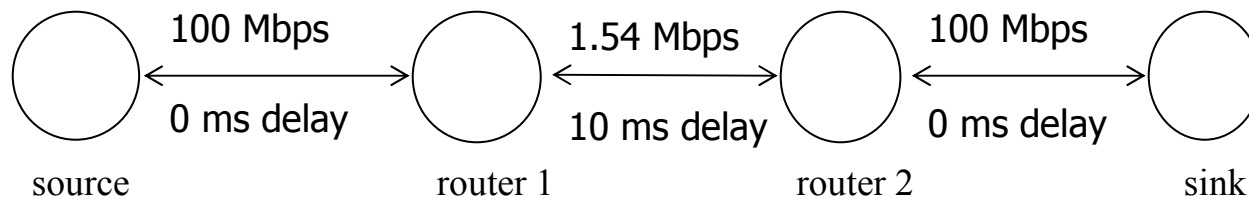


TCP/RED model

- Dynamical model of TCP/RED:

$$\bar{q}_{k+1} = \begin{cases} (1 - w_q) \cdot \bar{q}_k + w_q \cdot \max\left(\frac{N \cdot K}{\sqrt{p_k}} - \frac{C \cdot d}{M}, 0\right) & \text{if } p_k \neq 0 \\ (1 - w_q) \cdot \bar{q}_k + w_q \cdot \left(rwnd \cdot N - \frac{C \cdot d}{M}\right) & \text{if } p_k = 0 \end{cases}$$

Validation: simulation scenario



- source to router1:
 - link capacity: 100 Mbps with 0 ms delay
- router 1 to router 2: the only bottleneck in the network
 - link capacity: 1.54 Mbps with 10 ms delay
- router 2 to sink:
 - link capacity: 100 Mbps with 0 ms delay



RED: default parameters

- RED parameters:

S. Floyd, "RED: Discussions of Setting Parameters," Nov. 1997:
<http://www.icir.org/floyd/REDparameters.txt>

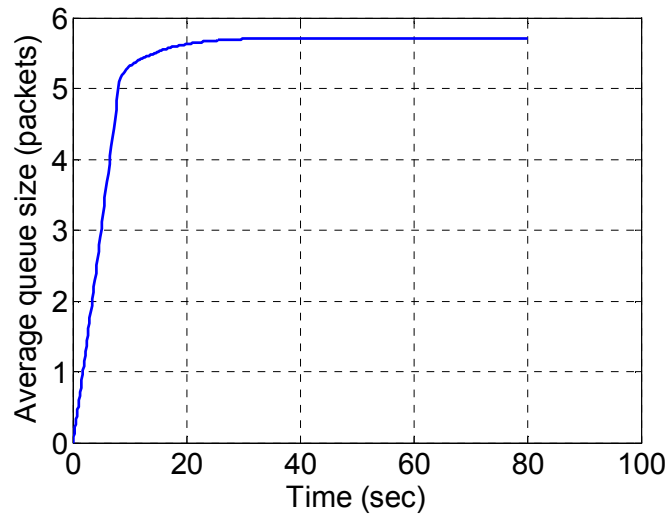
Queue weight (w_q)	0.002
Maximum drop probability (p_{\max})	0.1
Minimum queue threshold (q_{\min})	5 (packets)
Maximum queue threshold (q_{\max})	15 (packets)
Packet size (M)	4,000 (bytes)



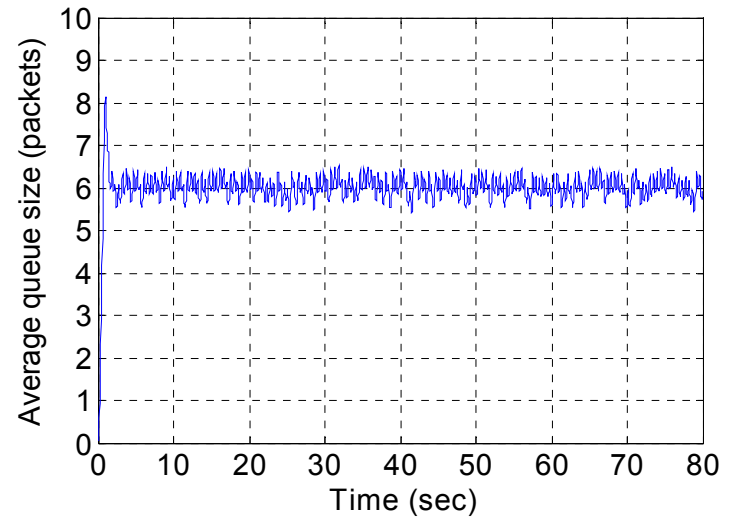
TCP/RED model validation

- Waveforms of the state variable with default parameters:
 - average queue size
- Validation for various values of the system parameters:
 - queue weight: w_q
 - maximum drop probability: p_{\max}
 - queue thresholds: q_{\min} and q_{\max} , $q_{\max}/q_{\min} = 3$

Average queue size: waveforms



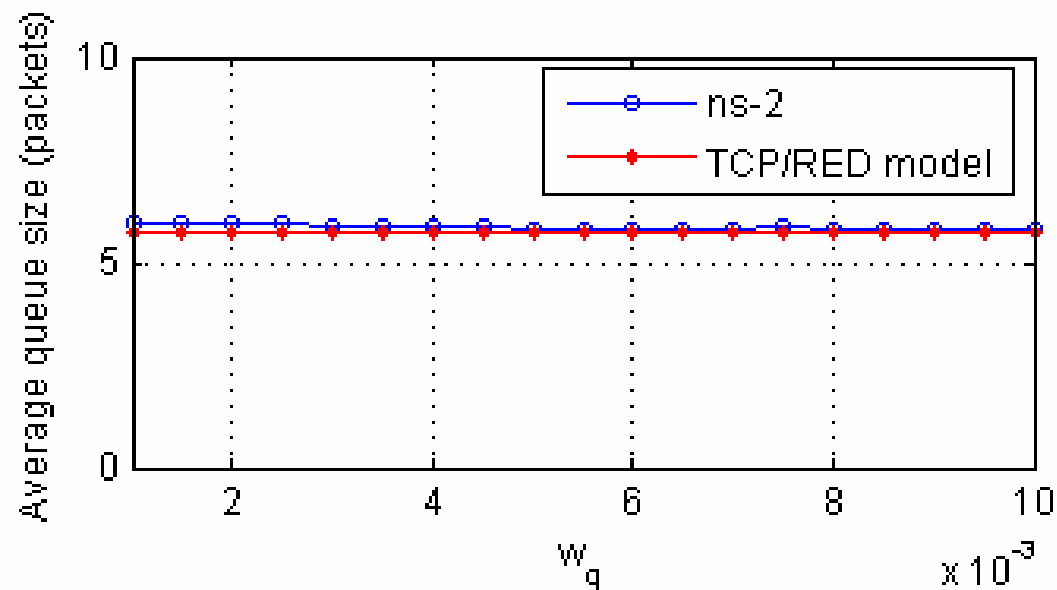
TCP/RED model



ns-2

Model validation: w_q

- average queue size during steady state:





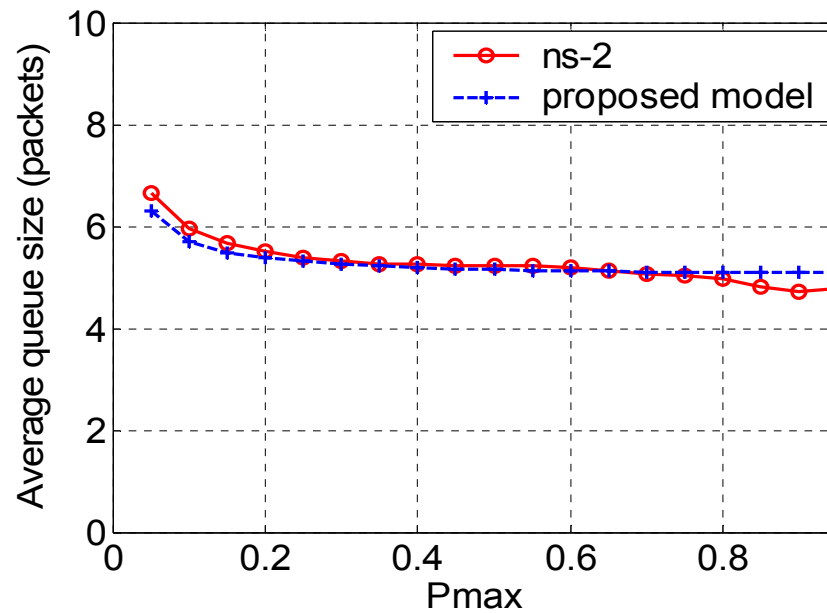
Model validation: w_q

- Comparison of system variables:

Parameter	Average RTT (msec)		Sending rate (packets/sec)		Drop rate (%)	
	TCP/RED model	ns-2	TCP/RED model	ns-2	TCP/RED model	ns-2
weight (w_q)						
0.001	164.6	36.1	385.6950	384.710	0.0421	0.543
0.002	143.8	36.0	385.7317	384.767	0.0356	0.546
0.004	137.1	36.2	385.3205	384.789	0.0486	0.556
0.006	135.2	35.8	385.4833	384.726	0.0486	0.556
0.008	134.7	35.8	385.5207	384.676	0.0486	0.549
0.01	134.6	35.7	385.5913	384.700	0.0483	0.546

Model validation: p_{\max}

- average queue size during steady state:





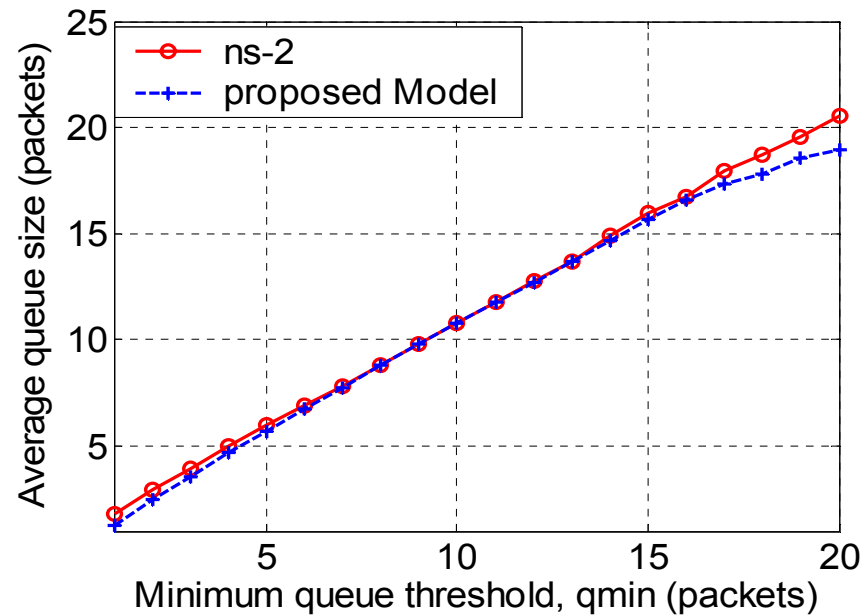
Model validation: p_{\max}

- Comparison of system variables:

Parameter	Average RTT (msec)		Sending rate (packets/sec)		Drop rate (%)	
	TCP/RED model	ns-2	TCP/RED model	ns-2	TCP/RED model	ns-2
p_{\max}						
0.05	161.7	38.1	385.7815	384.700	0.0323	0.510
0.1	143.8	36.0	385.6950	384.767	0.0421	0.546
0.25	131.7	34.5	385.4579	384.726	0.0518	0.585
0.5	126.9	34.0	385.5830	379.367	0.0551	0.613
0.75	125.9	35.1	385.3998	357.550	0.0572	0.647

Model validation: q_{\min} and q_{\max}

- average queue size during steady state:





Model validation: q_{\min} and q_{\max}

- Comparison of system variables:

q_{\min} (packets)	Average RTT (msec)		Sending rate (packets/sec)		Drop rate (%)	
	TCP/RED model	ns-2	TCP/RED model	ns-2	TCP/RED model	ns-2
3	103.6	31.1	385.2706	382.437	0.0875	0.709
5	143.8	36.0	385.6950	384.767	0.4210	0.546
10	238.8	48.1	385.7833	384.850	0.1300	0.331
15	307.9	60.3	386.9869	384.830	0.3216	0.224
20	343.8	73.0	387.9538	384.950	0	0.159

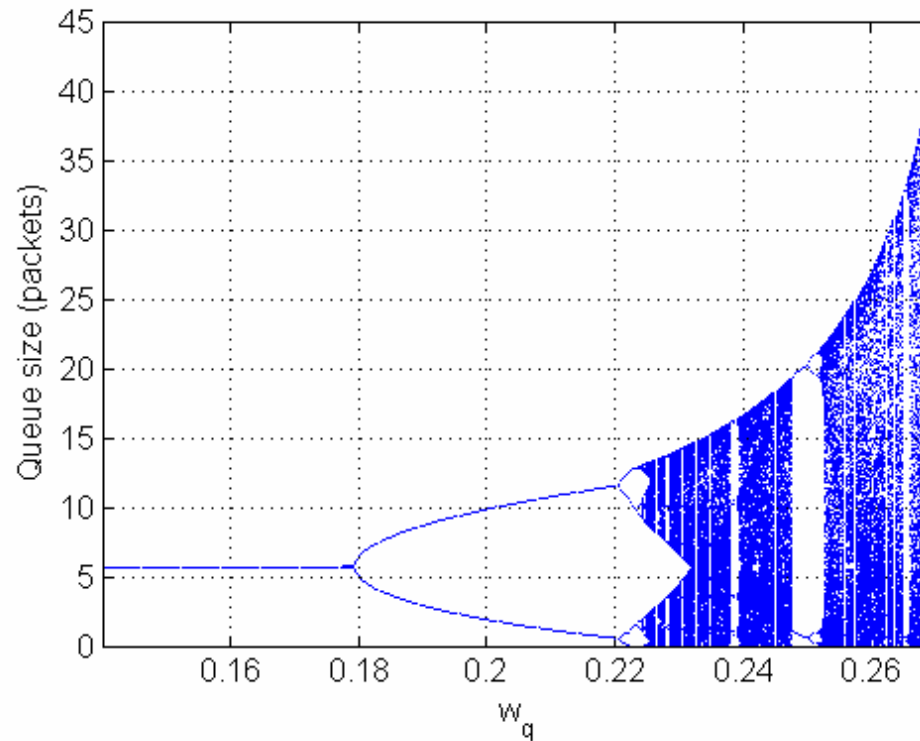


TCP/RED: model evaluation

- Waveforms of the **average queue size**:
 - match the ns-2 simulation results
- **Sending rate**:
 - reasonable agreement with ns-2 simulation results
- **Average RTT and drop rate**:
 - disagreement with ns-2 simulation results

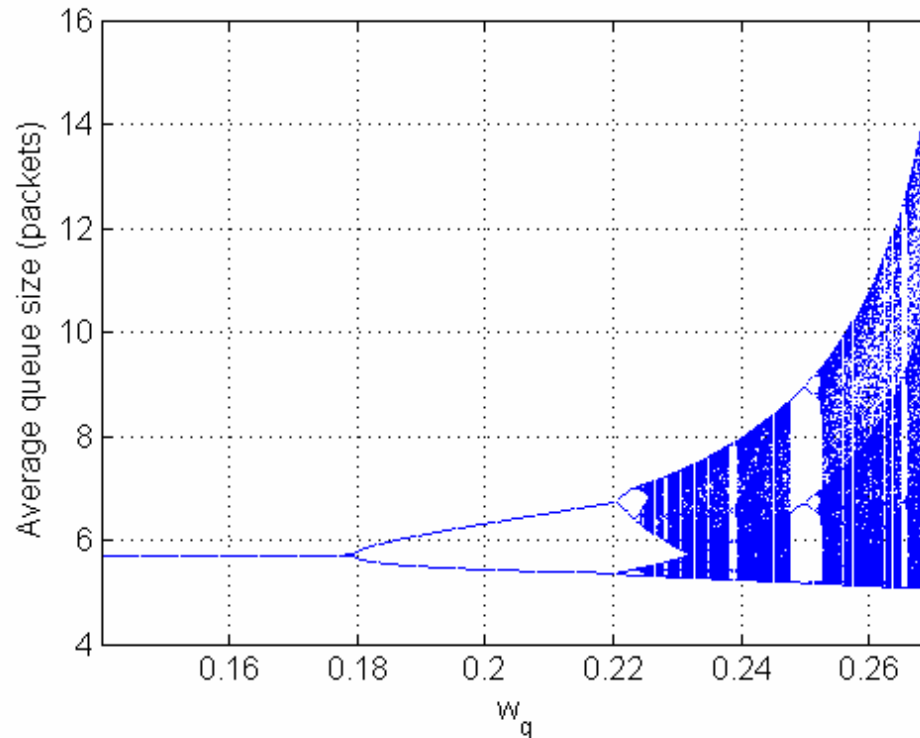
Queue size vs. w_q

- $p_{\max} = 0.1, q_{\min} = 5, q_{\max} = 15$



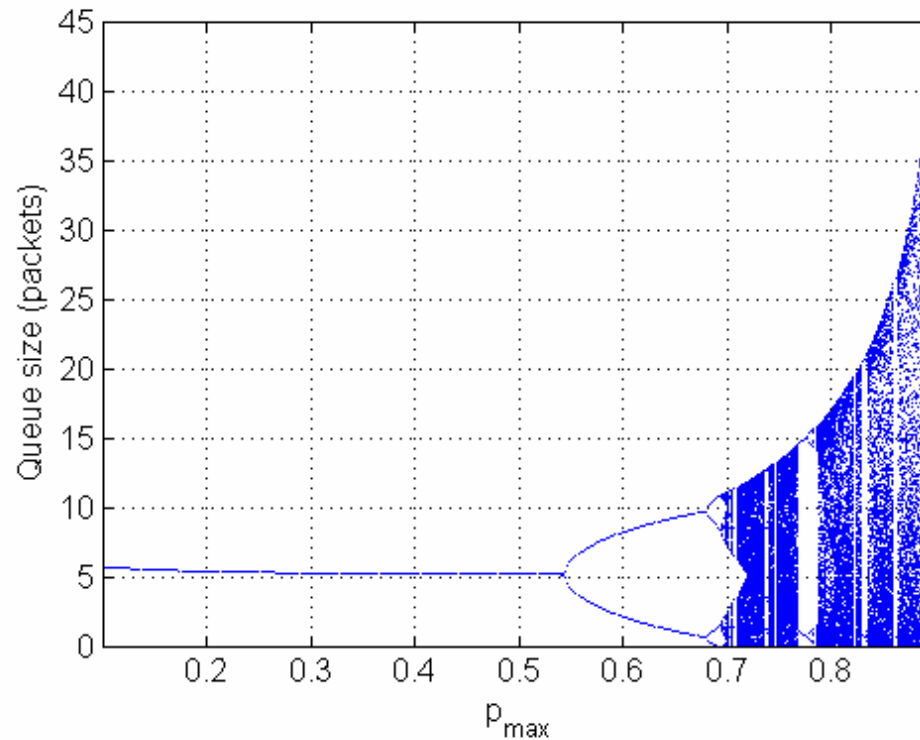
Average queue size vs. w_q

- $p_{\max} = 0.1, q_{\min} = 5, q_{\max} = 15$



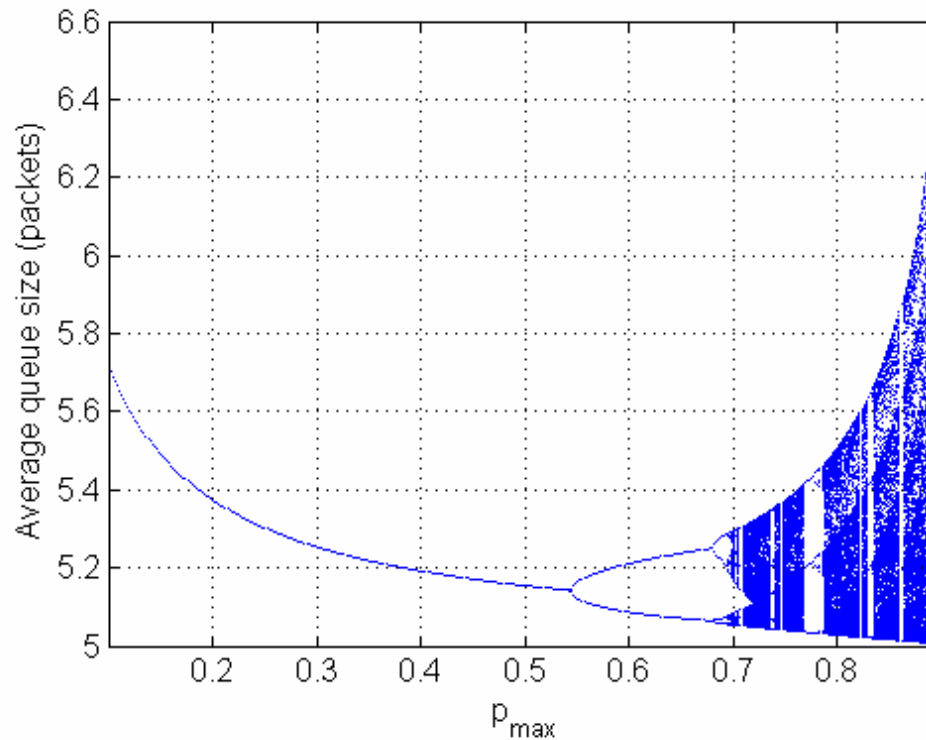
Queue size vs. ρ_{\max}

- $w_{q_x} = 0.04, q_{\min} = 5, q_{\max} = 15$



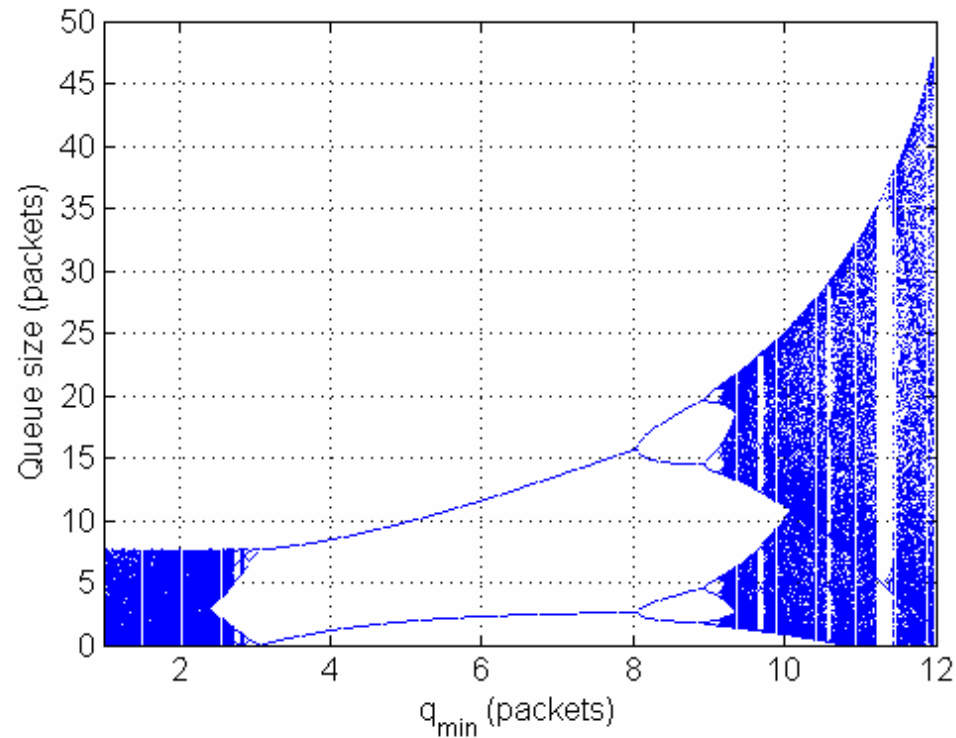
Average queue size vs. ρ_{\max}

- $w_q = 0.04, q_{\min} = 5, q_{\max} = 15$



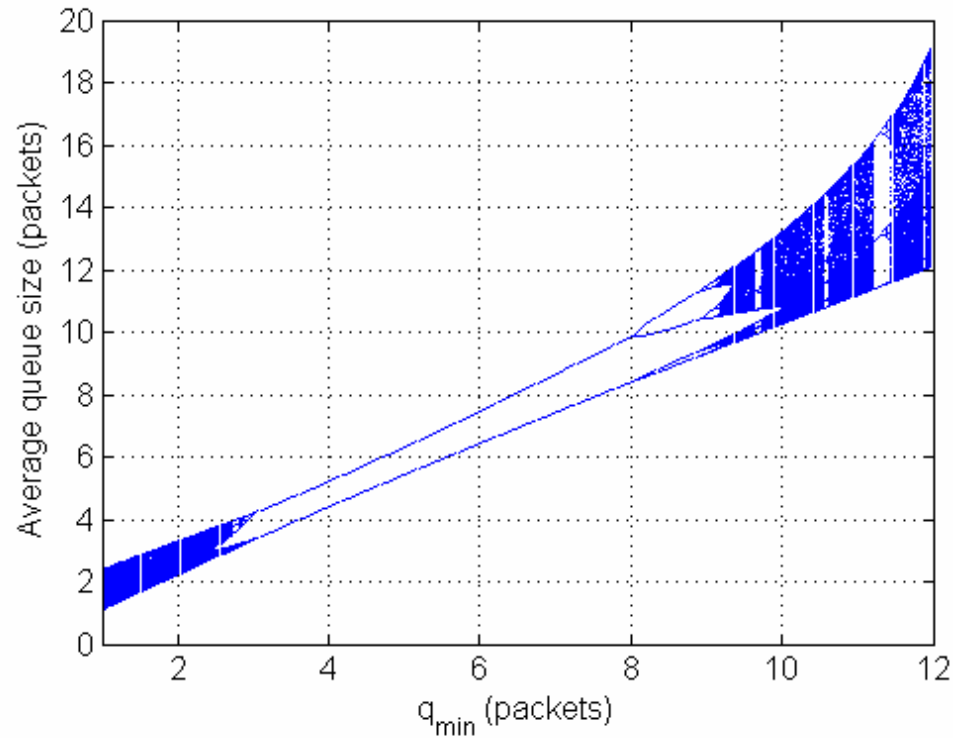
Queue size vs. q_{\min}/q_{\max}

- $w_q = 0.2, p_{\max} = 0.1, q_{\max} = 3 \times q_{\min}$



Average queue size vs. q_{\min}/q_{\max}

- $w_q = 0.2, p_{\max} = 0.1, q_{\max} = 3 \times q_{\min}$





Conclusion

- We developed a discrete-time model for TCP Reno with RED
- TCP/RED model include:
 - slow start, congestion avoidance, fast retransmit, timeout, elements of fast recovery, and RED
- Proposed model was validated by comparing its performance with ns-2 simulation results
- It captures the main features of the dynamical behavior of TCP Reno with RED
- The model was used to study bifurcation and chaos in TPC/RED systems with a single connection



References: TCP and RED

- [1] M. Allman, V. Paxson, and W. Steven, "TCP congestion control," *IETF Request for Comments (RFC) 2581*, Apr. 1999.
- [2] B. Barden et al., "Recommendations on queue management and congestion avoidance in the Internet," *IETF Request for Comments (RFC) 2309*, Apr. 1998.
- [3] R. Braden, "Requirements for Internet hosts: communication layers," *IETF Request for Comments (RFC) 1122*, Oct. 1989.
- [4] C. Casetti, M. Gerla, S. Lee, S. Mascolo, and M. Sanadidi, "TCP with faster recovery," in *Proc. MILCOM 2000*, Los Angeles, CA, USA, Oct. 2000, vol. 1, pp. 320–324.
- [5] K. Fall and S. Floyd, "Simulation-based comparison of Tahoe, Reno, and SACK TCP," *ACM Communication Review*, vol. 26, no. 3, pp. 5–21, July 1996.
- [6] V. Jacobson, "Congestion avoidance and control," *ACM Computer Communication Review*, vol. 18, no. 4, pp. 314–329, Aug. 1988.
- [7] V. Jacobson, "Modified TCP congestion avoidance algorithm," <ftp://ftp.isi.edu/end2end/end2end-interest-1990.mail>, Apr. 1990.



References: TCP and RED

- [8] J. Padhye and S. Floyd, "On inferring TCP behavior," in *Proc. ACM SIGCOMM 2001*, San Diego, CA, USA, Aug. 2001, pp. 287–298.
- [9] V. Paxson and M. Allman, "Computing TCP's retransmission timer," *IETF Request for Comments (RFC) 2988*, Nov. 2000.
- [10] J. Postel, "Transmission control protocol," *IETF Request for Comments (RFC) 793*, Sept. 1981.
- [11] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The protocols*. New York, NY: Addison-Wesley, 1994.

- [12] S. Floyd, "RED: discussions of setting parameters," Nov. 1997:
<http://www.icir.org/floyd/REDparameters.txt>.
- [13] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.
- [14] S. Floyd, R. Gummadi, and S. Shenker, "Adaptive RED: an algorithm for increasing the robustness of RED's active queue management," Aug. 2001:
<http://www.icir.org/floyd/papers/>.



References: TCP/RED models

- [15] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP latency," in *Proc. IEEE INFOCOM 2000*, Tel Aviv, Israel, Mar. 2000, vol. 3, pp. 1742–1751.
- [16] V. Firoiu and M. Borden, "A study of active queue management for congestion control," in *Proc. IEEE INFOCOM 2000*, Tel-Aviv, Israel, Mar. 2000, vol. 3, pp. 1435–1444.
- [17] J. Padhye, V. Firoiu, and D. F. Towsley, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Trans. Networking*, vol. 8, no. 2, pp. 133–145, Apr. 2000.
- [18] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, "A control theoretic analysis of RED," in *Proc. IEEE INFOCOM 2001*, Anchorage, AK, Apr. 2001, vol. 3, pp. 1510–1519.
- [19] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, "Analysis and design of controllers for AQM routers supporting TCP flows," *IEEE Trans. on Automatic Control*, vol. 47, no. 6, pp. 945–959, June 2002.



References: TCP/RED models

- [20] P. Ranjan, E. H. Abed, and R. J. La, "Nonlinear instabilities in TCP- RED," in *Proc. IEEE INFOCOM 2002*, New York, NY, USA, June 2002, vol. 1, pp. 249–258.
- [21] R. J. La, P. Ranjan, and E. H. Abed, "Nonlinearity of TCP and instability with RED," in *Proc. SPIE ITCOM*, Boston, MA, USA, July 2002, pp. 283–294.
- [22] P. Ranjan, R. J. La, and E. H. Abed, "Bifurcations of TCP and UDP traces under RED," in *Proc. 10 th Mediterranean Conference on Control and Automation (MED) 2002*, Lisbon, Portugal, July 2002.
- [23] P. Ranjan, E. H. Abed, and R. J. La, "Nonlinear instabilities in TCP-RED," *IEEE/ACM Trans. on Networking*, vol. 12, no. 6, pp. 1079–1092, Dec. 2004.
- [24] I. Khalifa and Lj. Trajković, "An overview and comparison of analytical TCP models," in *Proc. IEEE International Symposium on Circuits and Systems*, Vancouver, BC, Canada, May 2004, vol. V, pp. 469–472.