RED with Dynamic Thresholds for Improved Fairness

Vladimir Vukadinović and Ljiljana Trajković
{vladimir, ljilja}@cs.sfu.ca
Communication Networks Laboratory, Simon Fraser University, Vancouver, BC, Canada

Fair Active Queue Management

- Transmission Control Protocol (TCP) traffic represents vast majority of Internet traffic. TCP flows are responsive because TCP's congestion avoidance algorithm adapts their sending rates based on congestion conditions in the network.
- User Datagram Protocol (UDP) is the most commonly used protocol for real-time services. Its flows are unresponsive because UDP does not react to network congestion.
- TCP senders recognize lost packets as a sign of network congestion and reduces their sending rate. In contrast, UDP senders do not have any knowledge of congestion and their sending rates will not be adjusted.
- When responsive and unresponsive flows compete for the same output link in a router, unresponsive flows tend to occupy more than their fair share of the link capacity.
- One approach to solving the unfairness problem is to employ Active Queue Management (AQM) algorithms.
- Random Early Detection (RED) [1], the most widely implemented AQM algorithm, is unable to restrict unresponsive flows because it discards incoming packets from all flows with equal probabilities.
- Certain AQM algorithms, such as Flow Random Early Detection (FRED) [2], Longest Queue Drop (LQD) [3], and CHOKe [4], tend to identify and restrict unresponsive flows by preferentially discarding packets from these flows.
- We introduce a new AQM algorithm, named Random Early Detection with Dynamic Thresholds (RED-DT), that dynamically adapts queue parameters to achieve a more fair distribution of the link capacity.

RED-DT Algorithm

- In order to identify unresponsive and greedy flows, RED-DT maintains per-flow state for active flows. A flow is considered to be active if it has at least one packet in the queue.
- For each active flow, there is an entry in a flow table that contains instantaneous queue size q^i, average queue size q^i_{ave} , and maximum drop probability p^i_{max} . As in RED, instantaneous aggregate queue size q and average aggregate queue size q_{ave} are also monitored.
- Similar to RED, RED-DT maintains minimum and maximum queue thresholds, where $min_{th} = 3max_{th}$. However, in RED-DT these thresholds are dynamically adapted upon each packet arrival. If the new packet belongs to flow i, max_{th} is recalculated as:

$$max_{th} = (1 - \alpha\, p^i_{max})(B - q),$$

where B is the buffer size, p^i_{max} is the maximum drop probability for flow i, and α is a constant parameter. RED-DT calculates the drop probability for arriving packets as:

$$p = \begin{cases} 0 & \text{if } q^i_{ave} \le min_{th} \\ \dfrac{q^i_{ave} - min_{th}}{max_{th} - min_{th}}\, p^i_{max} & \text{if } min_{th} < q^i_{ave} < max_{thr} \\ 1 & \text{if } q^i_{ave} \ge max_{th} \end{cases}$$

- Unlike RED, which compares the average aggregate queue size q_{ave} with thresholds, RED-DT compares the average queue size of a particular flow q^i_{ave} with thresholds. When a packet is admitted to the queue, maximum drop probabilities p^i_{max} are updated as:

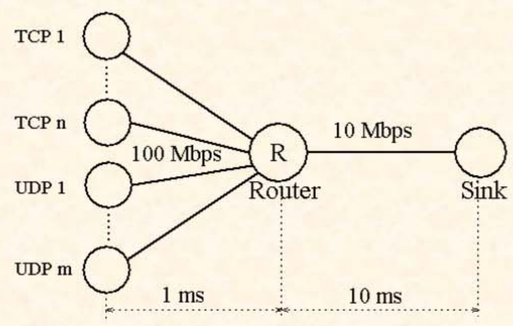
$$p^i_{max} = \begin{cases} max(1, p^i_{max} + \delta) & \text{if } q^i_{ave} > \dfrac{q_{ave}}{N} \\ p^i_{max} & \text{if } q^i_{ave} = \dfrac{q_{ave}}{N}, \\ min(p_{min}, p^i_{max} - \delta) & \text{if } q^i_{ave} < \dfrac{q_{ave}}{N} \end{cases}$$

where δ is a constant increment, p_{min} is RED-defined parameter, and N is the number of active flows.

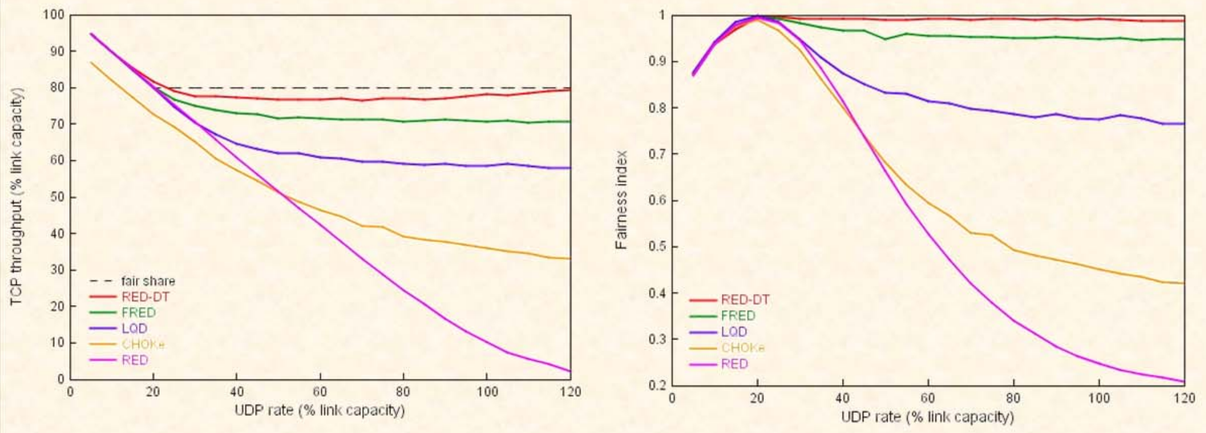
- In order to provide early feedback to responsive flows, the maximum drop probability cannot be smaller than p_{min}. One choice of the increment δ is $\delta = p_{min}$
- Maximum drop probability gradually increases for the flows whose average queue size is larger than $1/N$ of the average aggregate queue size q_{ave}. These flows are identified as potentially unresponsive (greedy) and their thresholds are decreased. When the average buffer occupancy of these flows decreases below q_{ave}/N, their maximum drop probability gradually decreases and their thresholds increase. This mechanism aims to distribute the buffer space equally among active flows.
- Recalculation of thresholds is the major complexity issue in RED-DT. Memory requirements for storing the per-flow states are limited to the size of the active flow table.

Simulation Results

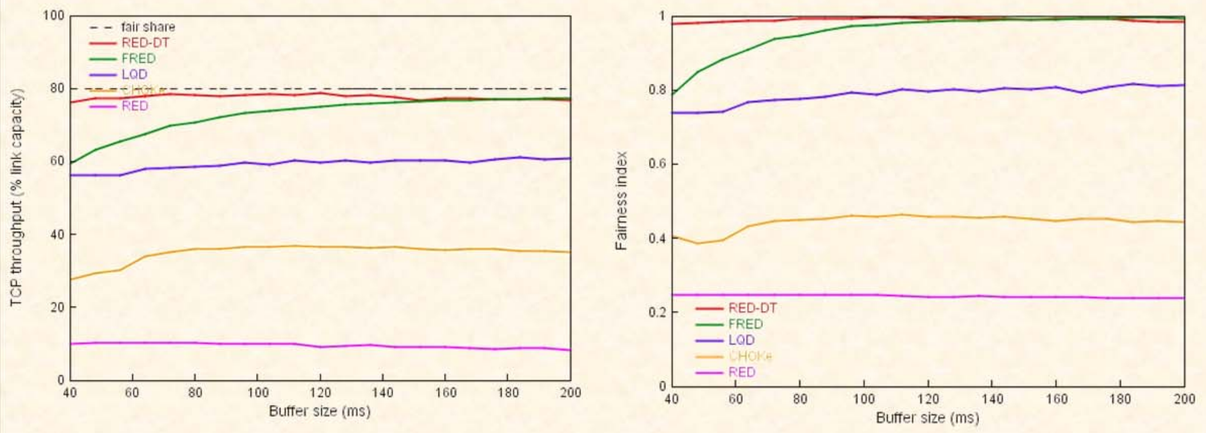
- We used ns-2 network simulator [5] to evaluate RED-DT and compare its performance to RED [1], FRED [2], LQD [3], and CHOKe [4].
- All simulation scenarios employ a common network topology with 16 TCP senders, 4 UDP senders, and a single bottleneck link.
- To transfer FTP data, TCP New Reno senders use fixed packet size of 500 bytes. UDP senders use 500-byte packets to transfer constant bit-rate streams.
- Default buffer size is 200 packets (80 ms of buffering on a 10 Mbps link).
- Minimum and maximum queue thresholds for RED, FRED, and CHOKe are 1/4 and 3/4 of the buffer size, respectively.
- UDP rate is 2.5 Mbps, except for the scenario with variable traffic load.
- We calculated Jain's fairness index [6] from the average throughput achieved by TCP and UDP flows. The values of the fairness index closer to 1 indicate a higher degree of fairness.



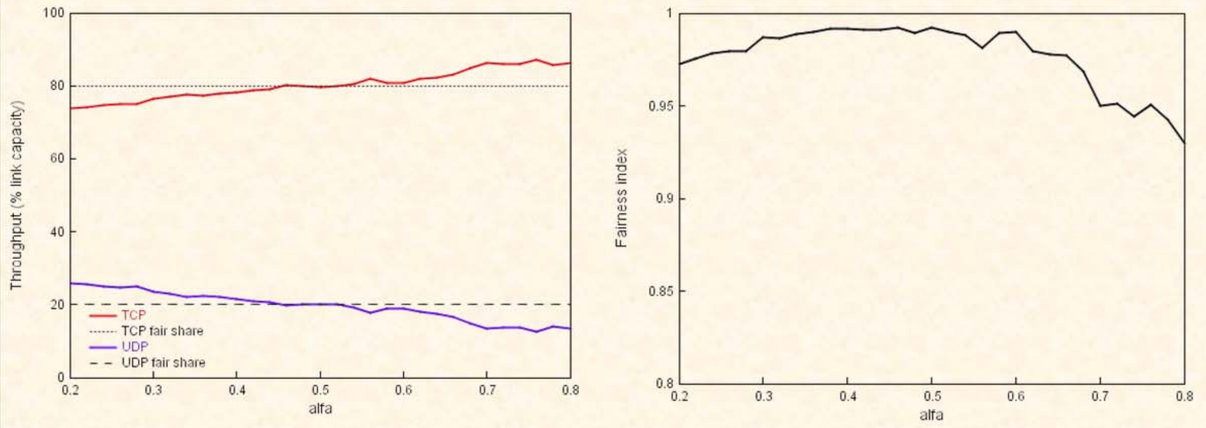
Influence of the traffic load:



Influence of the buffer size:



Influence of the parameter α:



References:

[1] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," in *IEEE Transactions on Networking*, vol. 1, no. 4, pp. 397 - 413, Aug. 1993.
[2] D. Lin and R. Morris, "Dynamics of random early detection," in *Proc. of ACM SIGCOMM '97*, Cannes, France, Oct. 1997, pp. 127 - 137.
[3] B. Suter, T. V. Lakshman, D. Stiliadis, and A. Choudhury, "Design considerations for supporting TCP with per-flow queuing," in *Proc. of IEEE INFOCOM '98*, San Francisco, CA, Apr. 1998, pp. 299 - 306.
[4] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe: a stateless active queue manage*ment* scheme for approximating fair bandwidth allocation," in *Proc. of IEEE INFOCOM '00*, Tel Aviv, Israel, Apr. 2000, pp. 942 - 951.
[5] The Network Simulator - ns-2: http://www.isi.edu/nsnam/ns/.
[6] R. Jain, *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation and modeling*. New York: John Wiley & Sons, 1991.