TCP and Network Traffic Self-Similarity: Causes and Remedies

K. Chandrayana *
Department of ECSE
Rensselaer Polytechnic Institute

*Work done with B. Sikdar, K. Vastola and S. Kalyanaraman

Overview

- Second order scaling: Definitions
- TCP and self-similarity: Causes
- TCP and self-similarity: Remedies
- Results
- Conclusions and Discussion

Self-Similar Nature of Network Traffic

- A rough definition of self similarity is that data looks "statistically similar" over a wide range of time scales
- Traffic "spikes" ride on longer term "ripples," that in turn ride on still longer term "swells" and so on

Self-Similarity: Definitions

- Number of different, not equivalent, definitions of self-similarity
- A continuous time process $Y = \{Y(t), t \in T\}$ is *self-similar* with self-similarity parameter H if

$$Y(t) \stackrel{d}{=} a^{-H} Y(at), \quad \forall t \in T, \quad \forall a > 0, \quad 0 \leq H < 1$$

- A stationary sequence $X = \{X(i), i \geq 1\}$ and

$$X^{(m)}(k) = 1/m \sum_{i=(k-1)m+1}^{km} X(i), k = 1, 2, \cdots$$

is exactly self-similar if for all m

$$X \stackrel{d}{=} m^{1-H} X^{(m)}$$

Implications of Self-Similarity

- Significant under-estimation of performance measures like packet delays or maximum queue size
- Larger congestion periods with heavily concentrated losses
- Linear increases in the buffer size do not result in large drops in packet drop rates
- Self-similar traffic with higher priority may starve streams with lower priority

Reducing the Scaling in Network Traffic

- Self-similarity leads to degraded network performance: large queue delays, large drop rates and long congestion periods
- The degradation is directly proportional to H
- No literature on solutions to reduce the degree of self-similarity in network traffic
- We investigate 2 methods to reduce scaling
- Methods based on eliminating TCP related causes

TCP and LRD

- Causes of self-similarity: Heavy tailed file size distribution, session level behavior, human think times
- Closed loop protocols lead to richer scaling
- TCP's congestion control and retransmission mechanisms also contribute to LRD in traffic
- Major contributors: Timeouts, exponential backoff
- The degree of heavy-tailedness depends on the loss rate

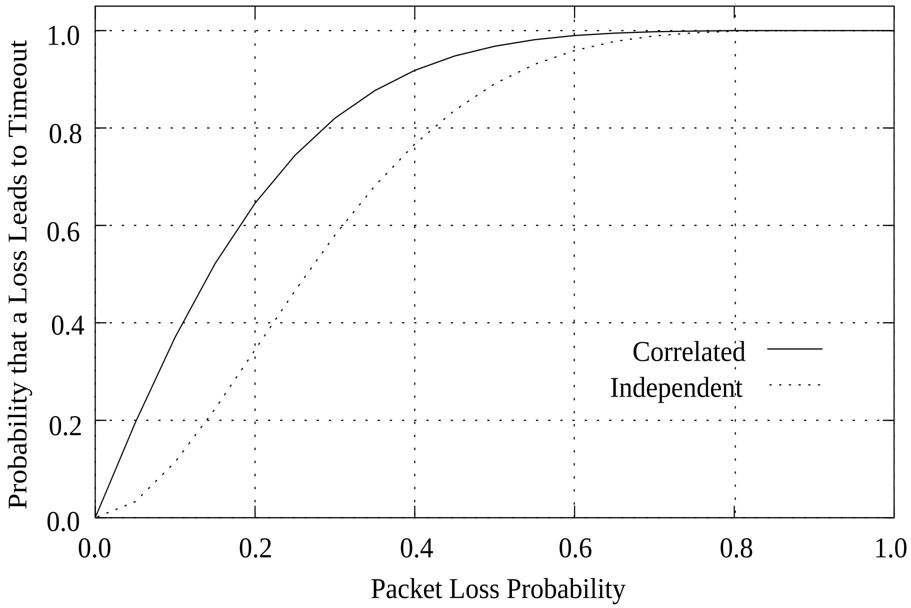
Causes and Solutions

- Causes
  - Timeouts and exponential backoffs
  - Burstiness and back to back packet transmission
  - ACK compression
- Solutions
  - Active queue management to reduce the number of timeouts in TCP flows
  - Pacing TCP packets to reduce burstiness and the effects of ACK compression

Buffer Management Policies and Self-Similarity

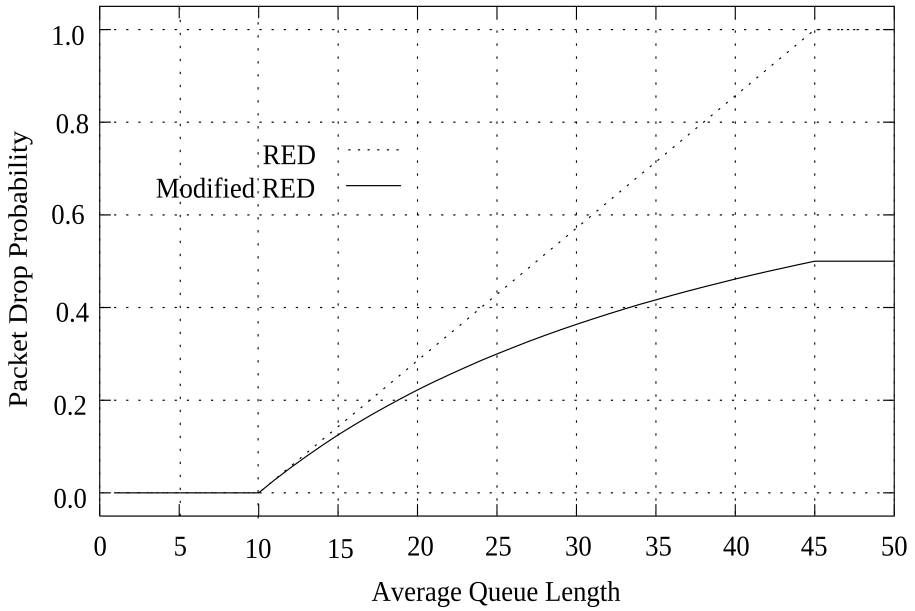
- Passive policies like Taildrop lead to correlated losses and higher degrees of self-similarity
- Active queue management policies like RED imply independent losses and thus lower timeouts
- At high loads, RED losses become fully correlated
- Modification to RED to eliminate consecutive losses reduces timeouts even at high losses

Loss and Timeout Probabilities



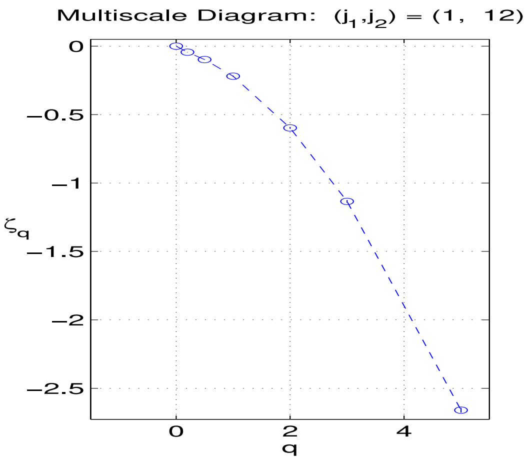
Probability of timeouts for TCP flows with $cwnd = 5$ with correlated and independent losses

Loss and Timeout Probabilities

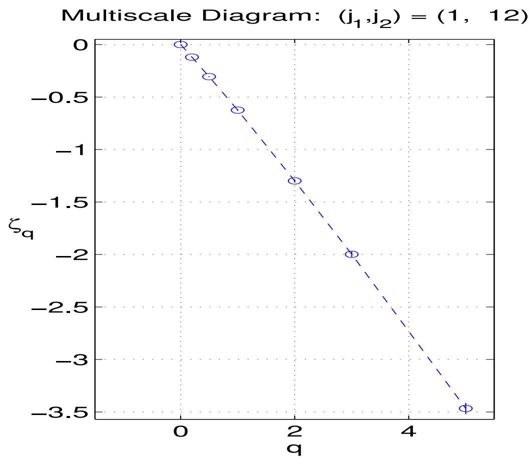


Packet drop probabilities in the RED and modified RED buffer management policies

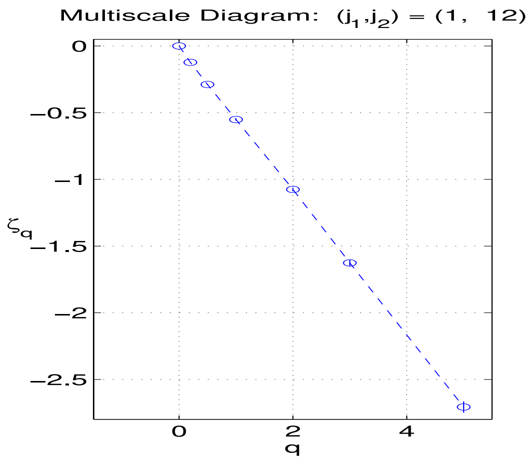
Results



Taildrop



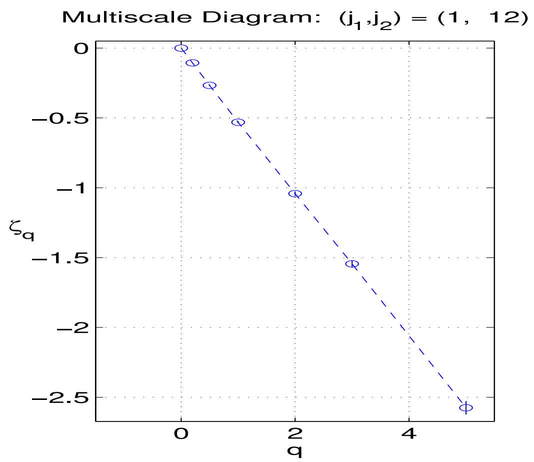
RED



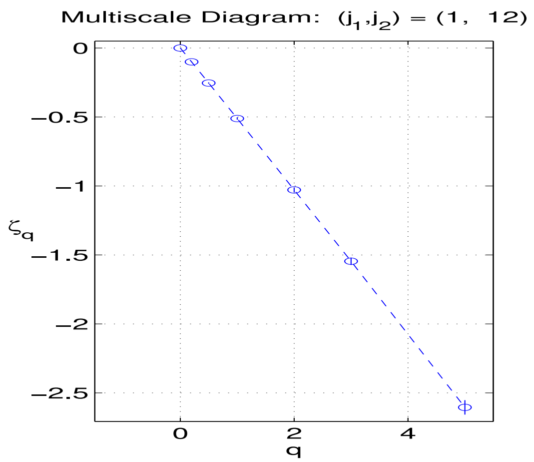
Modified RED

Scaling behavior without web traffic: 30 flows

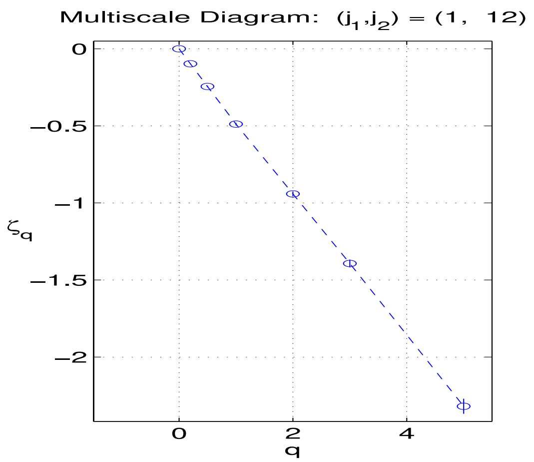
Results



Taildrop



RED



Modified RED

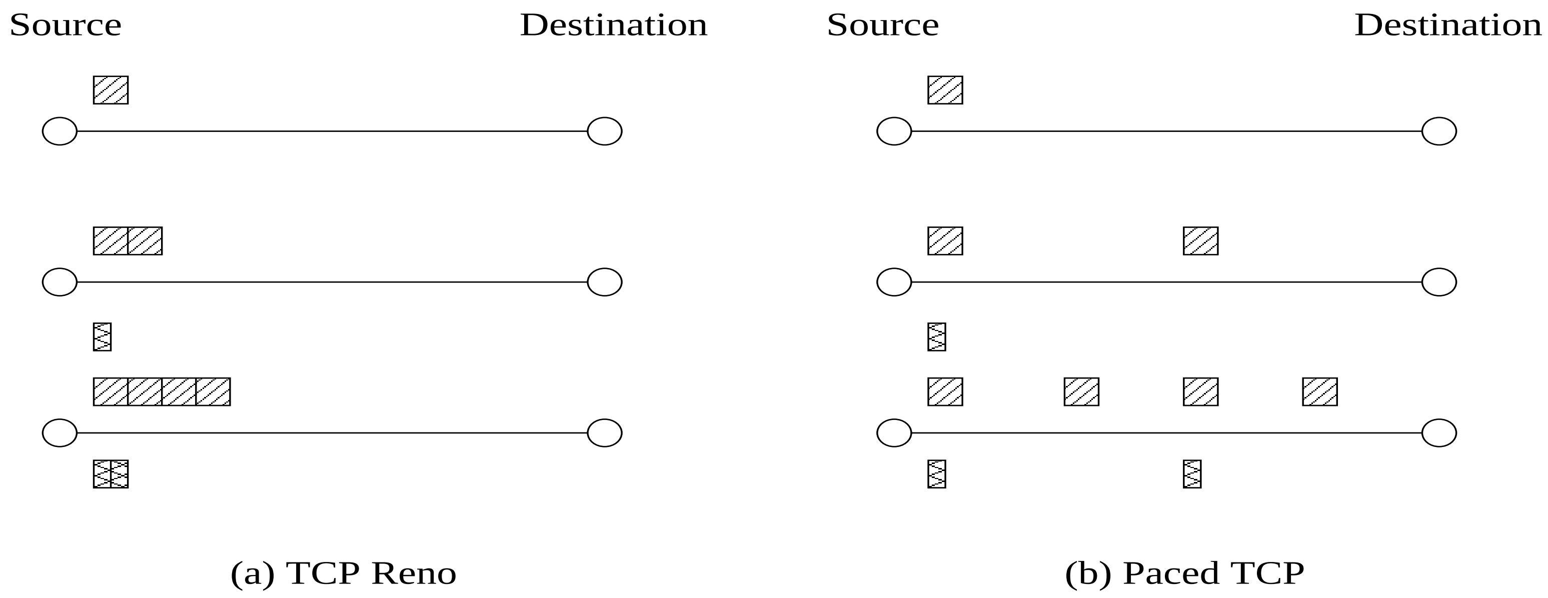
Scaling behavior with web traffic: 30 TCP flows and 10 web sessions

Numerical Results

Flows	Taildrop		RED		Modified RED	
	Throughput	H	Throughput	H	Throughput	H
10 L, 5 web	63988.44	0.58	72982.00	0.50	63055.77	0.50
15 L, 5 web	42099.70	0.67	44785.19	0.60	47422.37	0.57
10 L, 10 web	33005.78	0.74	37165.11	0.76	34517.33	0.75
15 L, 10 web	22318.52	0.77	23449.48	0.84	25204.88	0.77
20 L, 10 web	17508.89	0.80	19952.00	0.91	21275.88	0.80

Hurst parameter for queues with web traffic

Reducing Source-Level Burstiness of TCP Flows

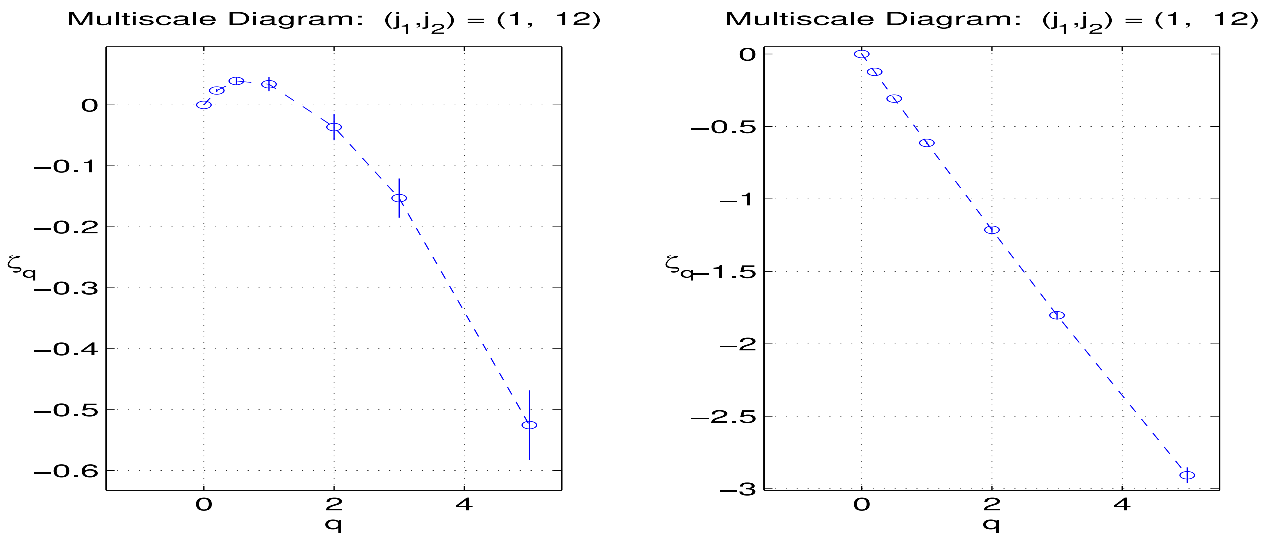


Packet sending pattern of Reno and paced TCP

Effects of Pacing

- TCP pacing removes burstiness and back to back packet transmissions at sources
- Pacing eliminates the harmful effects of ACK compression
- Delays between packets given by $\Delta = \frac{RTT}{cwnd}$

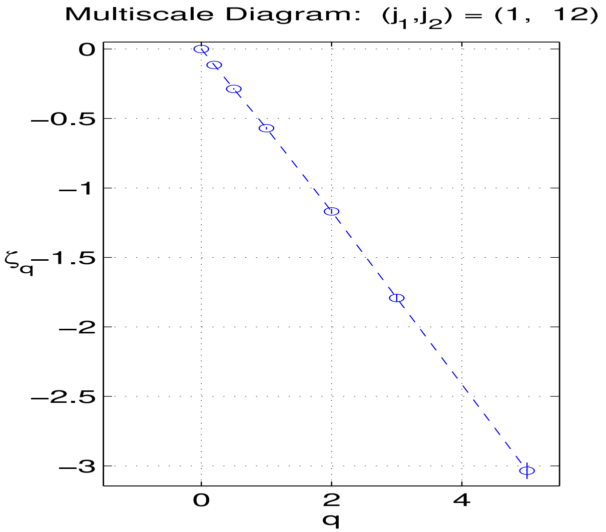
Results



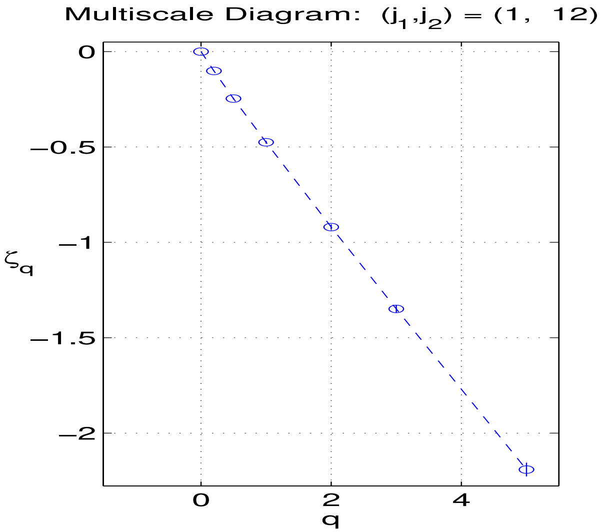
60 Flows, Reno   60 Flows, Paced

Scaling behavior without web traffic for Reno and Paced TCP: Taildrop queue

Results



40 Flows, Reno



40 Flows, Paced

Scaling behavior without web traffic for Reno and Paced TCP: RED queue

Numerical Results

Flows	Taildrop		RED	
	Reno	Paced	Reno	Paced
10 Long, 5 web	0.58	0.50	0.50	0.50
15 Long, 5 web	0.67	0.50	0.60	0.50
10 Long, 10 web	0.74	0.50	0.76	0.50
15 Long, 10 web	0.77	0.50	0.84	0.50
20 Long, 10 web	0.80	0.54	0.91	0.58

Hurst parameters for Reno and Paced TCP with web traffic for taildrop and RED queues.

Conclusions

- Reducing losses and ACK compression reduces scaling in network traffic
- Active queue management and TCP pacing can achieve these goals
- Taildrop queues lead to multi-fractal behavior while RED and modified RED cause mono-fractal scaling