An Improvement Mechanism for Low Priority Traffic TCP Performance in Strict Priority Queueing

Vahab Pournaghshband

Mahdi Rahimi

The Advanced Network and Security Research Laboratory

Computer Science Department

California State University, Northridge

Overview

Introduction

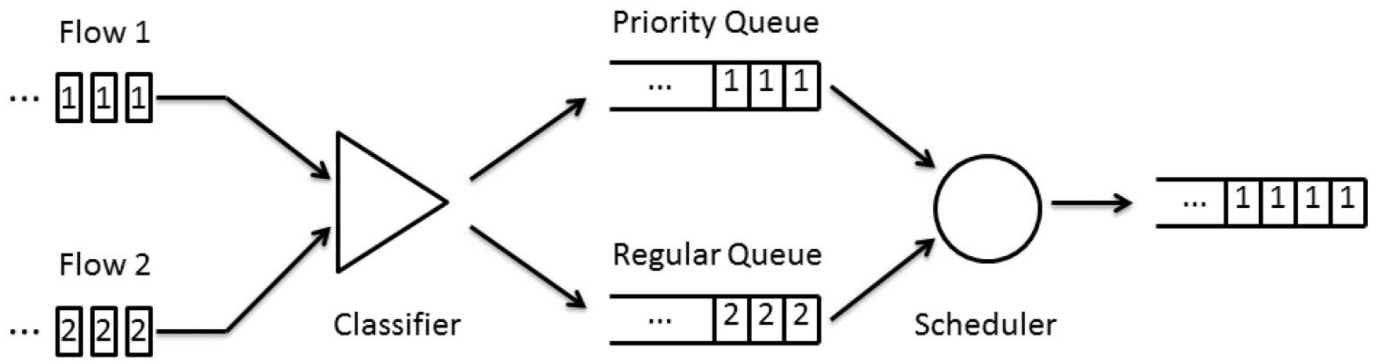
- A significant amount of business critical processes involve data transfers.
- Companies managing large volumes of data that must be exchanged with high frequency require reliable and quality data transfers, while sharing available network bandwidth with other applications.
    - Large corporations
    - Retailers
    - Publishers
    - Enterprises that
        - have large, central repository of assets.
        - produce high volumes of graphics, video, or data.
        - have multiple locations.
        - have high volume of international transfers.
        - timely distribution of files is business critical to them.
        - have distributed workgroups accessing the system.

Introduction

- ➢ Network Service Providers may treat customers differently or prioritize a group over others.
  - Internet Service Providers (ISPs)
  - Video-on-Demand providers
  - Cloud services and big data storage providers
- ➢ Other users may experience lower quality of service when the favored customers are being served.
- ➢ Motivates the companies into deploying a solution that alleviates this problem.

Introduction

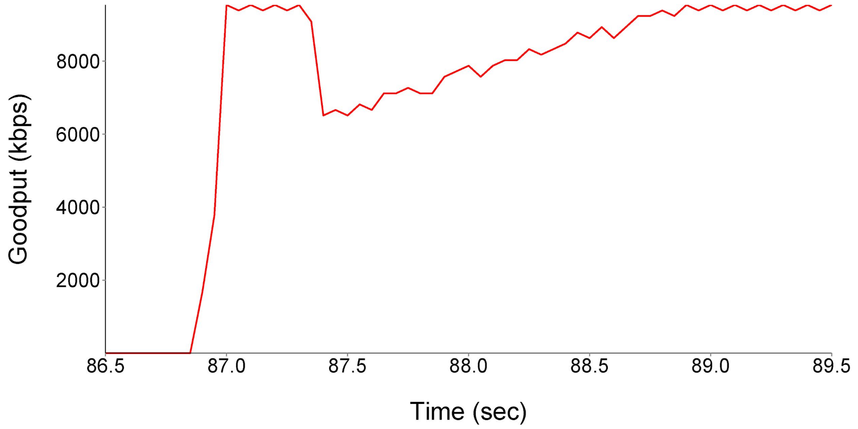
➢ Strict Priority Queuing (SPQ) is a widely used queuing method for applying preferential service on edge networks.



Packets from flow 2 cannot be sent until the priority queue is completely emptied of packets from flow 1

Introduction

- ➢ SPQ has a hostile nature toward lower priority flows.
- ➢ When the high priority traffic is present, the regular or low priority traffic is not served.
  - ○ The retransmission timeout of the low priority sender is expired.
  - ○ The sender feels congestion somewhere in the path.
  - ○ The congestion window size is dropped to its minimum value.
- ➢ When the high priority traffic is finished and the low priority sender resumes transmission of data, it enters the TCP slow-start state.
  - ○ Little by little, the sender increases its congestion window size until it finally reaches its previous amount.



Low priority traffic's slow-start state immediately after high priority queue is empty.

Introduction

- Furthermore, SPQ can cause termination of low priority connections.
  - TCP protocol defines a threshold for transmission of a certain packet.
  - When the high priority traffic is large enough to saturate the link and long enough to continue for a period of time, no low priority packet will have a chance to pass the SPQ.
  - The low priority sender retransmits the data for a period of time.
  - When the threshold is passed, it terminates the connection.

Methodology

- ➢ We propose *Amicable Strict Priority Queueing (ASPQ)*, an approach to address the shortcomings of SPQ and to improve the TCP performance of low priority traffic, in the presence of interruptions caused by a surge of high priority traffic flows.
- ➢ Our proposed approach has two major advantages.
  - First, it does not require any changes to the TCP stack of the end hosts.
  - Second, through the simple augmentation introduced in this paper, no modification to the existing SPQ-enabled switches and routers is required, making it readily deployable in current systems.
- ➢ Our approach is inspired by Freeze-TCP

Methodology

Freeze-TCP:

- ➢ Freeze-TCP is an end-to-end TCP enhancement mechanism for mobile environments.
- ➢ In mobile environments, temporary disconnections (due to signal fading or other link errors or due to the movements of mobile nodes) can cause packet loss, which in turn triggers TCP's slow start mechanism.
- ➢ Freeze-TCP uses the zero window advertisement and the persist mode capacity of TCP protocol to mitigate this problem.
- ➢ if a mobile receiver senses an impending disconnection, it will advertise a zero window advertisement on behalf of the receiver to force the sender into the persist mode and prevent it from reducing its congestion window size.

Methodology

TCP Persist Mode:

- ➢ In a regular TCP communication, the receiver may send a Zero Window Advertisement (ZWA) packet to the sender, stating that it cannot accept more packets.
- ➢ Upon receiving a ZWA packet, the sender enters a state called persist mode. In this state, the sender
  - ○ stops sending or retransmitting packets.
  - ○ periodically probes whether the receiver is ready again to accept packets, by sending Zero Window Probe (ZWP) packets.
  - ○ never terminates the connection as long as ZWPs are acknowledged.
  - ○ sender keeps its congestion window size intact
- ➢ When the receiver updates its window size to a value greater than zero, the sender resumes transmitting data.

Methodology

Amicable Strict Priority Queueing (ASPQ):

- A middlebox called *ASPQ controller* sits behind a regular SPQ-enabled device. The ASPQ controller
  - observes the traffic from both directions.
  - is aware of the existence of any high priority or low priority traffic at any time.
  - has an internal table that keeps a list of all of the existing high and low priority traffic.
- Upon the emergence of the first high priority flow, the ASPQ controller takes the low priority senders into the persist mode by advertising ZWAs to them. In order to do that, the ASPQ controller
  - captures the in-flight ACKs.
  - sets their window sizes to zero.
  - recalculates checksum.
  - then lets them pass.

Methodology

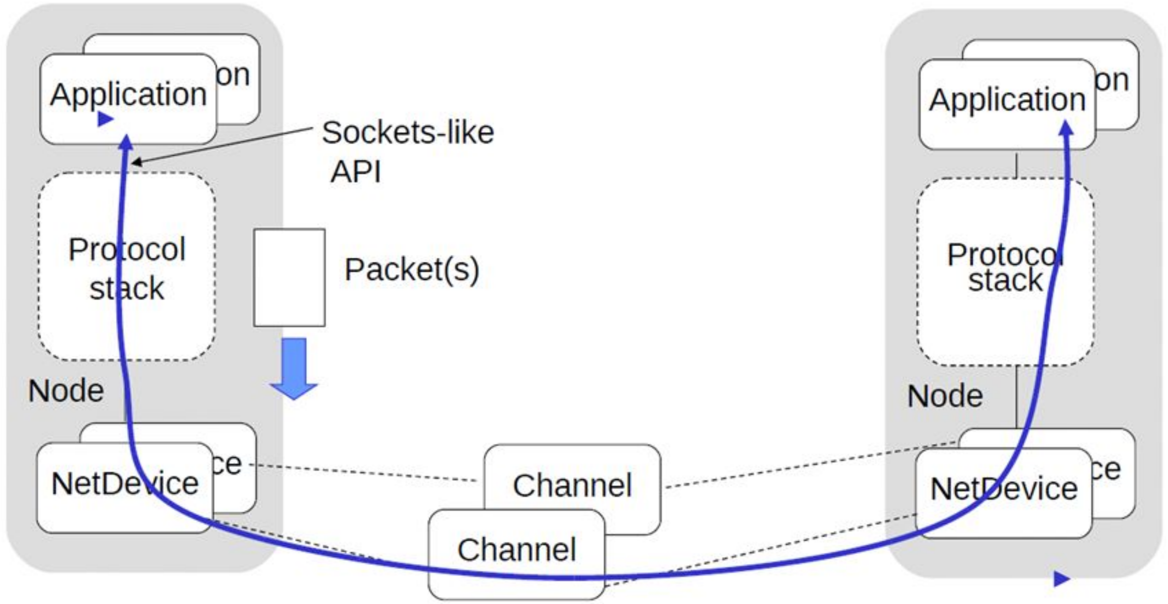
- By taking the senders into the persist mode
  - The congestion window sizes of the senders are kept intact.
  - The senders send ZWPs periodically, and the ASPQ controller acknowledges them on behalf of the receivers. Thus, the senders do not terminate the connection.
- When the the high priority traffic is finished, the ASPQ controller signals the low priority senders to resume the transmission.
  - The ASPQ controller always keeps a copy of the last ACK of each flow.
  - It sends the last ACK of the low priority flows to the senders.
  - The senders see the ACKs as a window update packet and resume the transmission.
  - Since they have kept their congestion window size intact, they resume with full speed achieving maximum possible channel utilization.

Network Simulator 3 (ns-3)

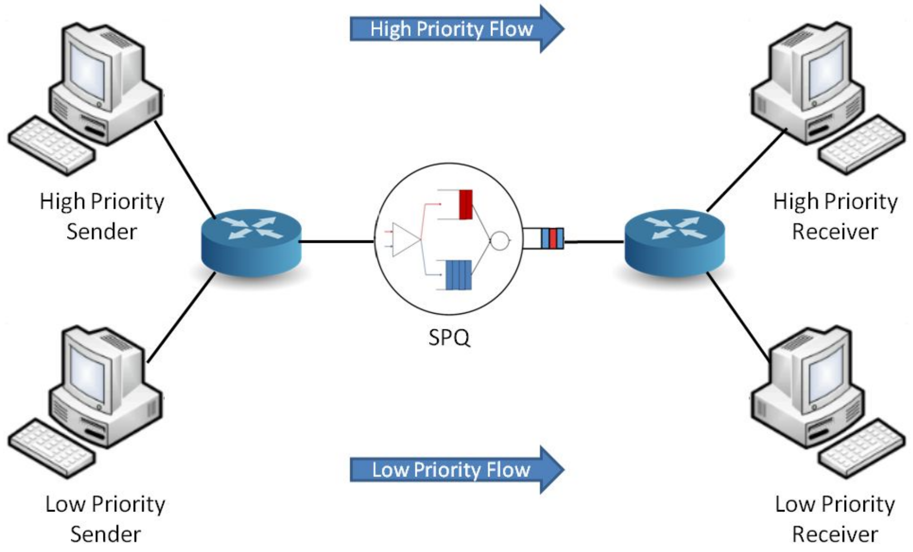
- A discrete-event open source simulator for networking research and education.
- Completely different from predecessor ns2, aiming to be easier to use and more ready for extension.
- ns3 core is written entirely in C++.
- Sophisticated simulation features included.
  - Extensive parameterization system
  - Configurable embedded tracing system, with standard outputs to text logs or PCAP (tcpdump/wireshark)
- Object oriented design for rapid coding and extension.
- Models true IP stack, with potentially multiple devices & IP addresses on every node.
- BSD lookalike, event based sockets API.

Network Simulator 3 (ns-3)

Looks just like IP architecture stack

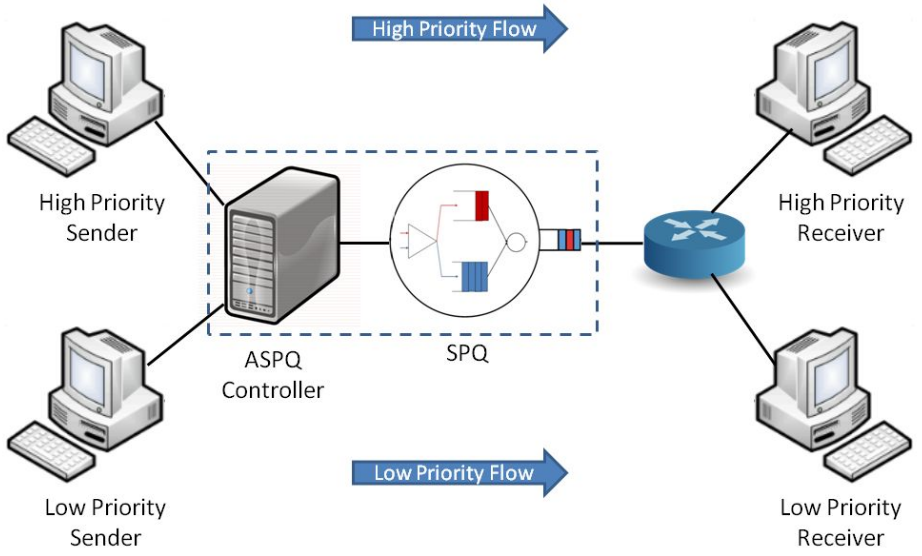


Evaluation Topology



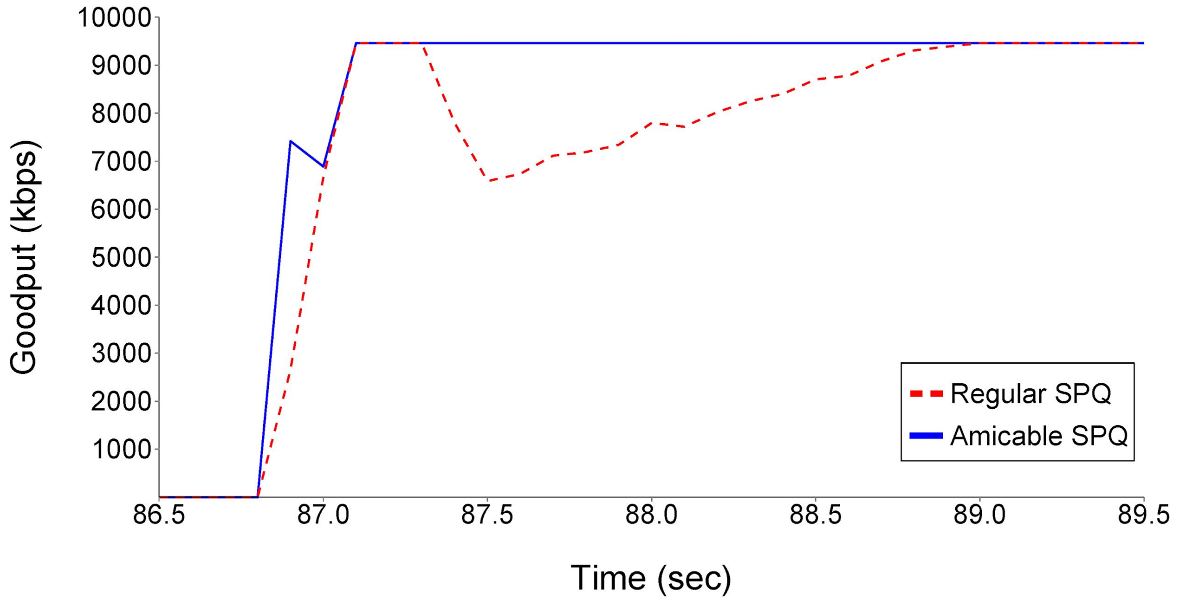
The SPQ experiments topology

Evaluation Topology



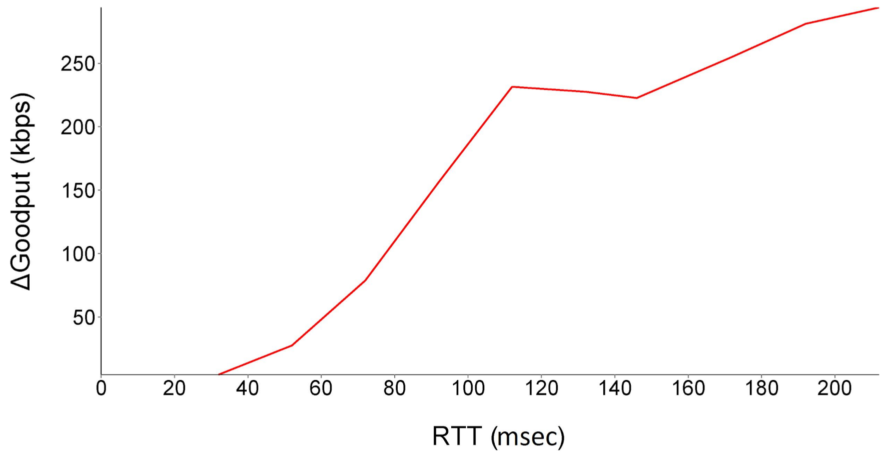
The ASPQ experiments topology

Evaluation Results

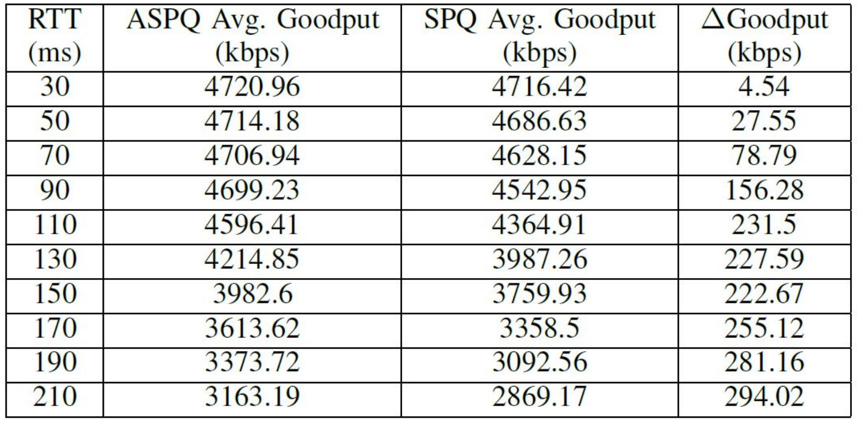


Comparison of the performance of ASPQ against SPQ

Evaluation Results



Measurement of ΔGoodput against different RTTs



RTT (ms)	ASPQ Avg. Goodput (kbps)	SPQ Avg. Goodput (kbps)	ΔGoodput (kbps)
30	4720.96	4716.42	4.54
50	4714.18	4686.63	27.55
70	4706.94	4628.15	78.79
90	4699.23	4542.95	156.28
110	4596.41	4364.91	231.5
130	4214.85	3987.26	227.59
150	3982.6	3759.93	222.67
170	3613.62	3358.5	255.12
190	3373.72	3092.56	281.16
210	3163.19	2869.17	294.02

Details of measurement of ΔGoodput against different RTTs

Summary and Future Work

- The shortcomings of Strict Priority Queuing
- A new approach to improve the TCP performance of low priority traffic in Strict Priority Queuing
- Future work:
  - Investigating how ASPQ acts in real networks by conducting a series of live experiments.
    - The ASPQ controller will be implemented using Click Modular Software Router.
    - A Cisco Catalyst 3750 switch will be used as a SPQ enabled middlebox.
    - PlanetLab nodes will be used as the senders and receivers.
  - Implementing our approach via Software Defined Networking (SDN).
    - A smart and central view of networks.
    - Porting the functionality of the ASPQ controller into a SDN controller.