Performance Evaluation of Scheduling Mechanisms for Broadband Networks

Gayathri Chandrasekaran

Master's Thesis Defense

The University of Kansas

07.31.2003

Committee:

Dr. David W. Petr (Chair)

Dr. Joseph B. Evans

Dr. David L. Andrews

Abbreviations

- HF^2Q - Hybrid FIFO Fair Queuing
- WFQ - Weighted Fair Queuing
- FIFO - First In First Out
- GPS - Generalized Processor Sharing
- BWA - Broadband Wireless Access
- DOCSIS - Data Over Cable Service Interface Specification
- MAC - Medium Access Control
- QoS - Quality of Service

Presentation Outline

- Introduction & Motivation
- Scheduling Mechanisms & Fair queuing
- Description of HF^2Q
- Performance Characterization of HF^2Q
- DOCSIS 1.1 Operation
- Performance Evaluation of DOCSIS 1.1
- Summary of Results
- Contributions & Future Work

Introduction & Motivation

- Future broadband networks will support multiple types of traffic over single physical infrastructure
- Issue of scheduling mechanisms and bandwidth allocation play a critical role
- Packet Fair Queuing algorithms have received a lot of attention
- Hybrid FIFO Fair Queuing (HF^2Q) – a new queuing discipline
  - Has very interesting properties and hence a performance characterization would prove worthwhile

Introduction & Motivation *(cont.)*

- DOCSIS 1.1 - de facto standard for delivering broadband services over HFC networks, defines MAC scheduling mechanisms for BWA, developed by CableLabs

- The IEEE 802.16 protocol was developed for Broadband Wireless Access (BWA) systems to provide Internet access and multimedia services to end users.
  - Consolidation of two proposals, one of which was based on DOCSIS 1.1

- DOCSIS 1.1 built on top of DOCSIS 1.0, specifies QoS components in terms of MAC scheduling based on underlying traffic requirements

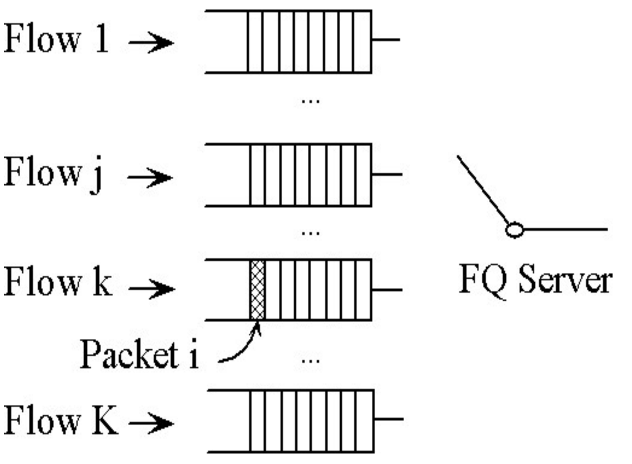
- Standardized in 2001 – a performance evaluation is significant in terms of understanding the operation of the protocol as such and its new QoS features

Scheduling Mechanisms and Fair Queuing

- Queuing algorithms determine the way packets from different sources interact
  - Controls order in which packets are sent
- In the classic FIFO the order of arrival completely determines bandwidth allocation
  - “Fair” in the sense that all packets get equal mean waiting times
- Protection against ill-behaved sources is required and hence the requirement of an algorithm that provides “fair” bandwidth allocation
- A class of fair queuing algorithms have been proposed since then

Background on FQ policies

- FQ algorithms maintain separate queues for packets from individual sources
- Queues are serviced in a round robin manner
- Provide **fair** treatment for supported flows by splitting bandwidth based on pre-defined weights
- Fairness Criterion : For any two backlogged flows, each flow's service (normalized to its weight) should be nearly the same

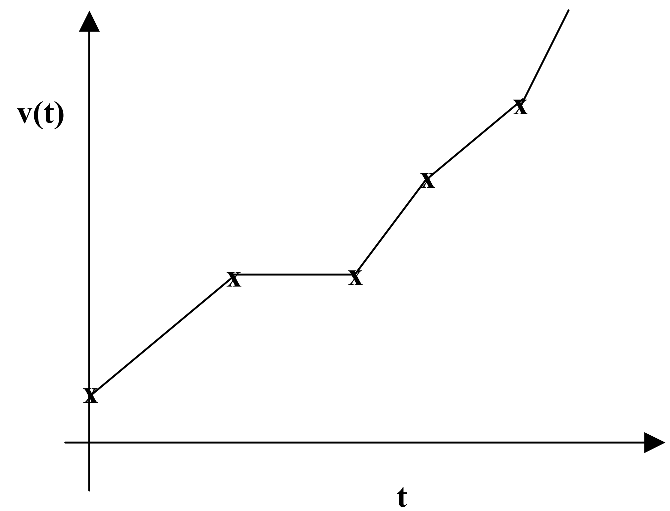


Weighted Fair Queuing

- WFQ is a packet scheduling technique allowing guaranteed bandwidth services
- WFQ is a GPS approximation for packet networks
  - GPS is an idealized fluid model that cannot be implemented practically
- The time at which a packet would complete service in GPS is computed and the packet is assigned a timestamp
  - Timestamps are **virtual finish times**
- Packets are served in increasing order of their timestamps

WFQ : Virtual time

- WFQ operation linked to the GPS system using virtual time
  - defines the order in which packets are served
- Virtual time *v(t)* is a piecewise linear increasing function of time
- Has a rate of increase inversely proportional to the sum of the service rates of the backlogged flows



$$v(t_2) - v(t_1) = \frac{C}{\sum_{k \in B(t_1,t_2)} r_k}.(t_2 - t_1)$$

WFQ : Virtual Start and Finish Times

- i[th] packet of flow k is denoted by p_k^i, its arrival time by a_k^i and its length by L_k^i
- $v(a_k^i)$ is the value of $v(t)$ at the time of packet arrival
- Virtual finish times are assigned as shown
- Virtual start time denoted by S_k^i varies depending on whether a new arrival is to an empty flow or backlogged flow
- WFQ serves packets in increasing order of finish times

$$F_k^i = \frac{L_k^i}{r_k} + S_k^i,$$
$$S_k^i = \max(F_k^{i-1}, v(a_k^i)),$$
$$k \in \mathbf{K}, \quad i = 1, 2, 3, \ldots$$

HF^{2}Q description

- A new and an interesting queuing algorithm discovered while developing the WFQ simulation model
- A small change in the WFQ operation leads to HF^{2}Q
- Named Hybrid FIFO Fair Queuing since it exhibits desirable properties of both FIFO and WFQ
- Exhibits FIFO behavior when the total system load is less than unity
- Behaves as WFQ when the system is overloaded

HF²Q implementation

- *v(t)* represented by *roundNumber*

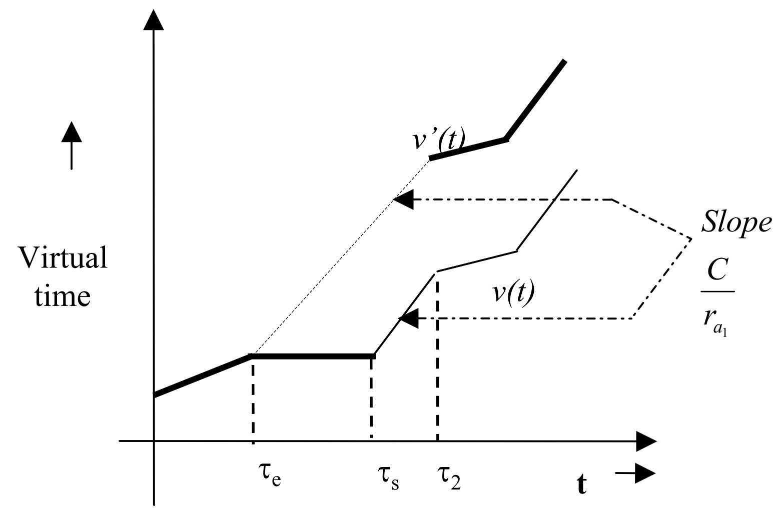
$$roundNumber = roundNumber + \frac{(currentTime - lastRUpdateTime)}{weightSum}.capacity$$

- Instants when the *roundNumber (v(t))* is updated is kept track of using a variable *lastRUpdateTime*

- The *lastRUpdateTime* is not updated when a new packet arrives after an idle period in HF²Q
- Scheduling order thus changes

HF^2Q vs. WFQ Virtual Time

- τ_e - end of a server busy period, τ_s- instant when the first packet arrives after an idle period from flow a_1, τ_2 - the instant of arrival of second packet
- Offset between *v(t)* and *v'(t)*
- The new arrival at τ_2 gets assigned *v'(t)* in HF^2Q rather than *v(t)* in WFQ

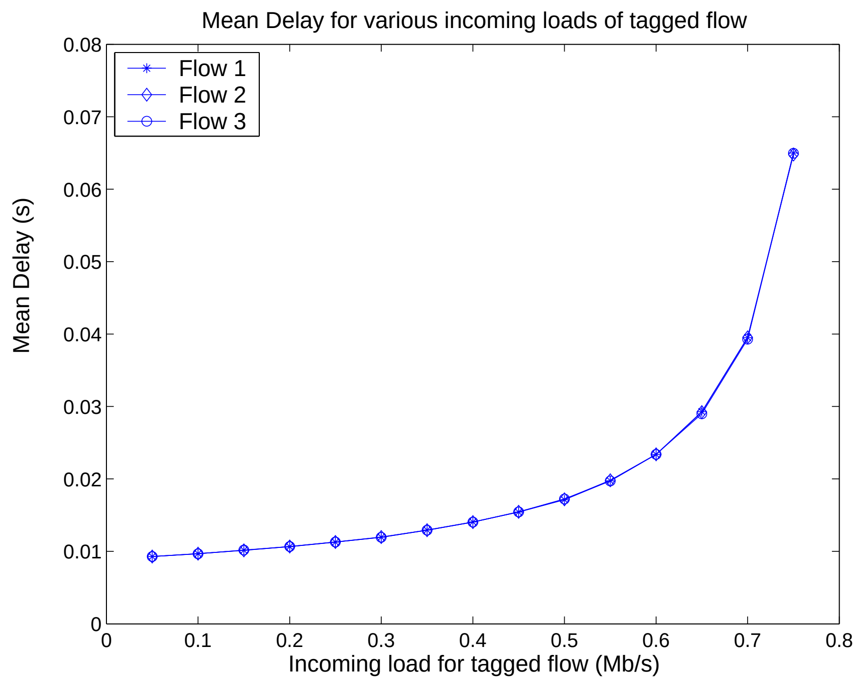


Performance Characterization

- Delay and throughput characteristics were studied and comparisons where made with WFQ
- Simulations were performed using Extend, using discrete event simulation techniques
- Link speed of 1Mbps, mean packet lengths of 8000 bits (uniformly distributed service times)
- Poisson arrivals with exponential inter-arrival times were used
- All experiments were performed with three flows, with flow 1 always being the *tagged* flow (i.e. the flow whose load is varied)

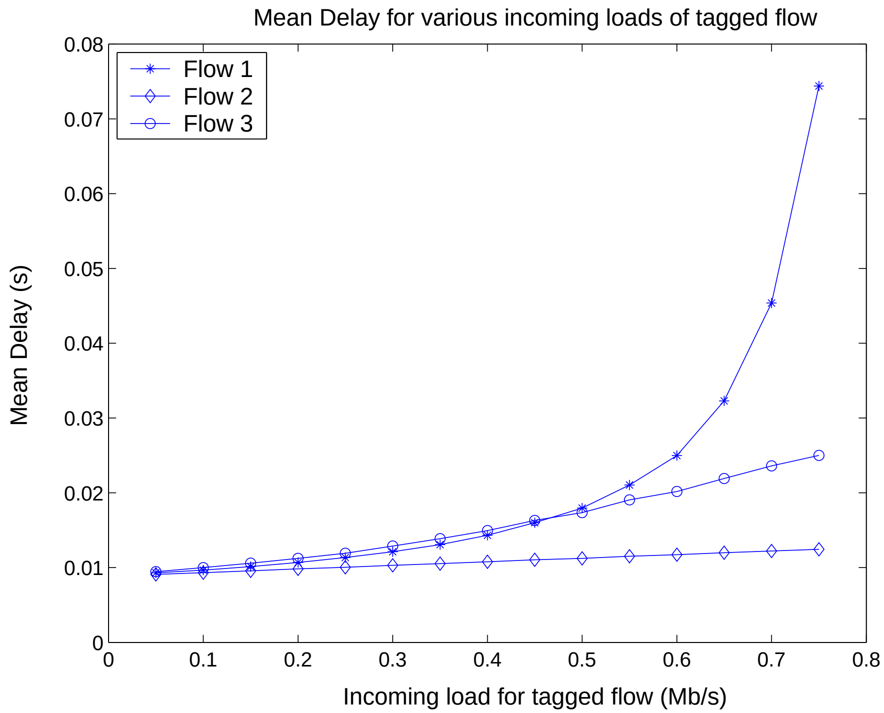
Delay Results :HF^{2}Q: Load < 1

- The three flows had reservations of 0.2, 0.7 & 0.1 Mbps respectively
- The tagged flow load was varied and the incoming loads of flows 2 & 3 were 0.09 Mbps
- The tagged flow exceeds its reservation after a load of 0.2 Mbps
- It is seen that HF^{2}Q behaves as FIFO offering equal mean waiting times for all flows



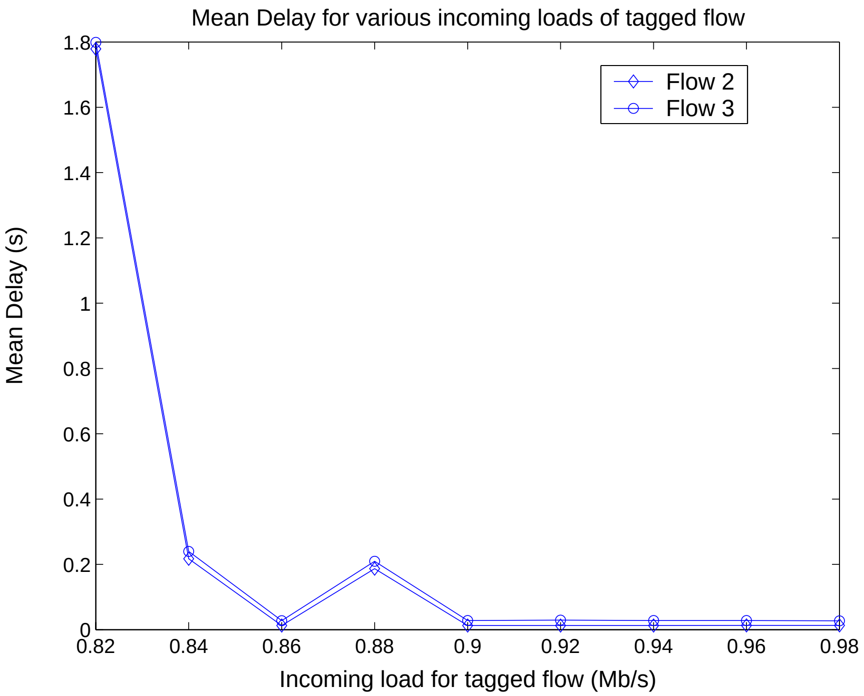
Delay Results :WFQ: Load < 1

- Same reservations and incoming loads were maintained for all flows
- It is observed that WFQ offers protection to the well-behaved flows (2 & 3)
- The delay of the tagged flow increases after it exceeds its reservation
- Flow 2 gets a lower delay compared to flow 3 since its reservation is higher



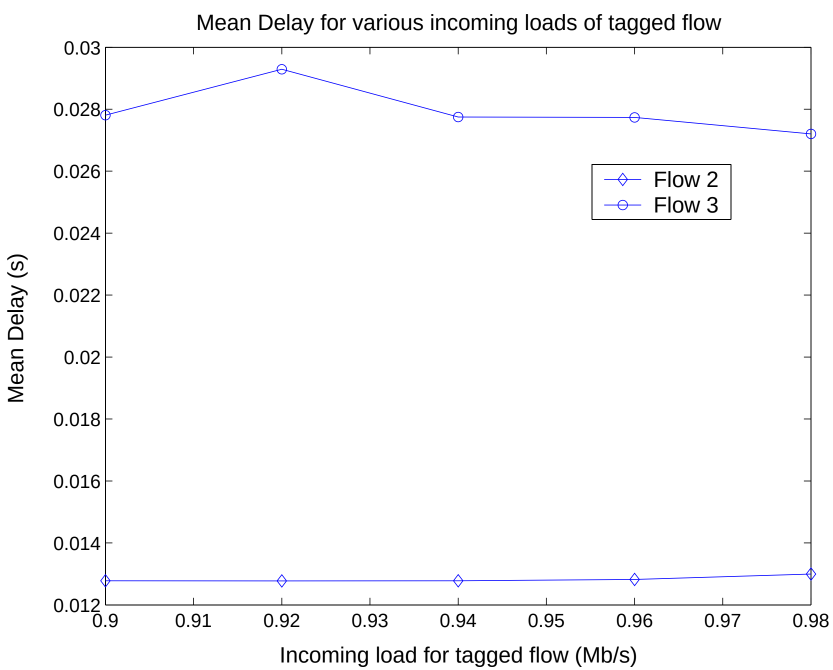
Delay Results :HF^{2}Q: Load > 1

- The tagged flow was increased so that the system load reaches 116%
- The delay of the tagged flow is not shown since it increases infinitely
- As is seen , the delay of the other two flows is low and bounded after a small transitional period
- As will be shown, HF^{2}Q exhibits WFQ behavior after the transitional period

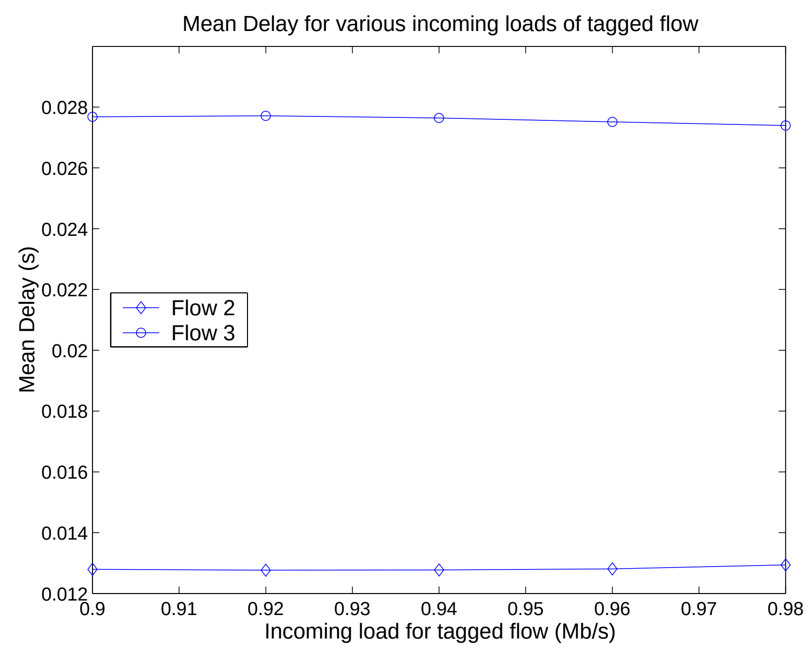


Delay Results (Load > 1 *cont.*)

- Result for HF^2Q (after the transitional region)



- Result for WFQ (after the transitional region)

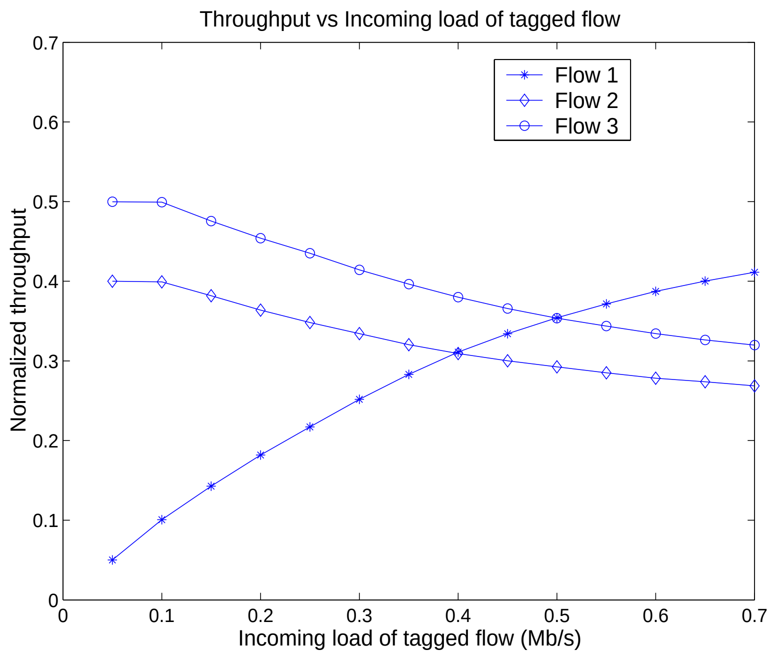


Throughput Experiment

- Throughput - fraction of link capacity used to carry packets
- Three flows with reservations of 0.5 , 0.3 & 0.2 Mbps
- Flows 2 & 3 had incoming loads of 0.4 & 0.5 Mpbs respectively
- Flows 2 & 3 heavily exceed their reservation
- The system is overloaded after the incoming load of the tagged flow exceeds 0.1 Mbps
- The throughput behavior of FIFO, HF^2Q and WFQ were studied

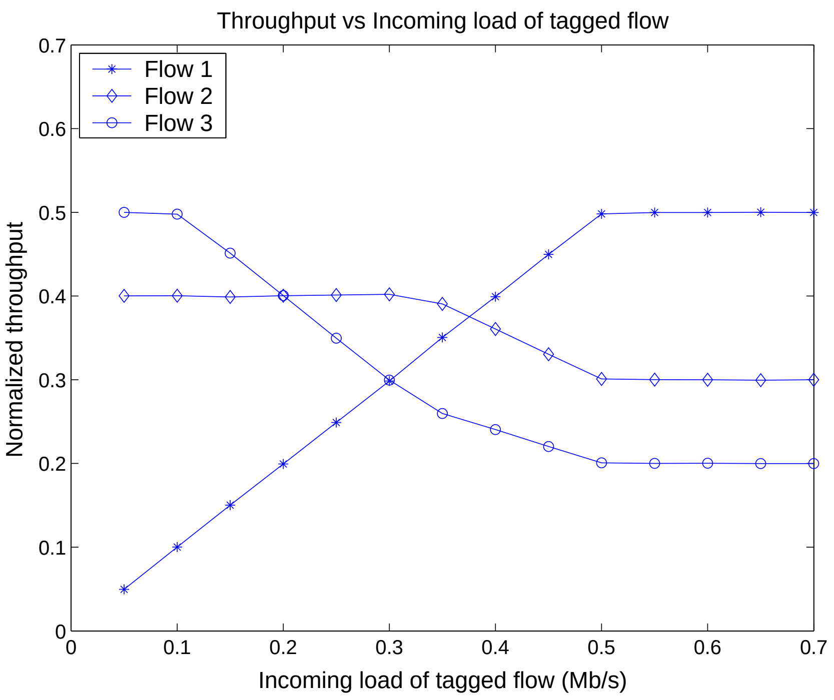
Throughput Results - FIFO

- All work conserving algorithms provide throughput equal to the incoming load when the total system load is less than unity
- When the system is overloaded FIFO offers throughput proportional to offered load
- WFQ and HF^2Q on the other hand protects the well-behaving flows

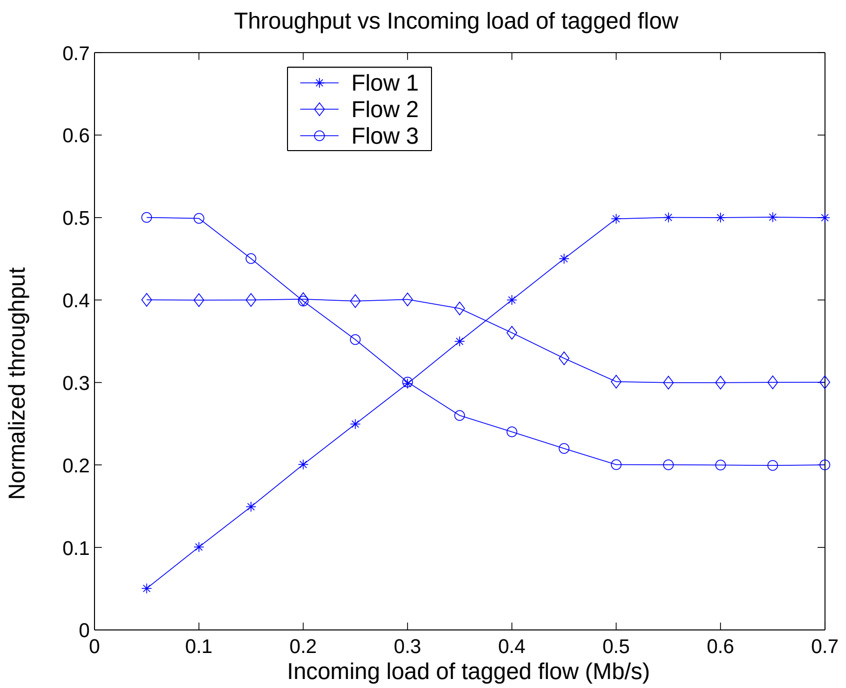


Throughput Results – HF²Q & WFQ

- HF²Q



- WFQ



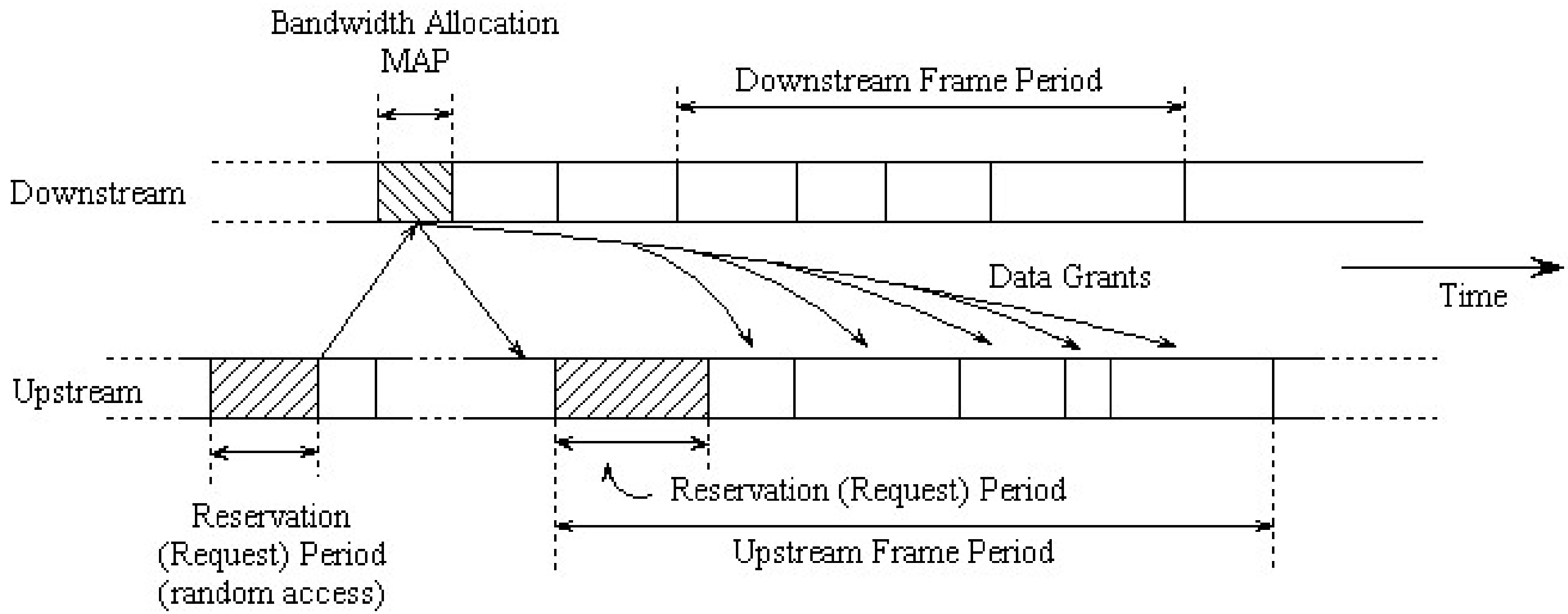
Conclusions Drawn

- HF^2Q services packets in FIFO order when the total system load is less than one
  - Indicated by equal mean waiting times
  - Verified via simulations with packet sequence numbers
  - Initial start-up period when ordering is not FIFO
- HF^2Q services packets as WFQ does when heavily overloaded
- Transitional period when the load is barely over unity

DOCSIS Introduction

- In DOCSIS a Cable Modem Termination System (CMTS) controls the operations of many terminating Cable Modems (CMs).
- Upstream and downstream channels are separated using FDD.
- Each upstream channel is further divided into a stream of fixed-size time minislots (TDMA).
- DOCSIS MAC utilizes a request/grant mechanism to coordinate transmission between multiple CMs.

DOCSIS Operation



Request Mechanisms

- Contention - the CM may transmit a request in the contention period
- Piggybacking - is a request for additional bandwidth sent in a data transmission
- Unsolicited grants – fixed size grants offered in a periodic basis
- Unicast request polls - unicast request opportunities are sent as a means of real-time polls regardless of network congestion

DOCSIS 1.1 QoS Classes

- Unsolicited Grant Service (UGS)
  - Offers fixed unsolicited size grants on periodic basis
  - Designed for fixed size data packet flows on fixed intervals
- Real Time Polling Service (rtPS)
  - Designed to support real-time flows that generate variable size packets on a periodic basis
  - Offers periodic unicast polls that allow CM to specify the size of the desired grant

DOCSIS 1.1 QoS Classes (*cont.*)

- Non Real Time Polling Service (nrtPS)
  - Designed to support non-real time flows that require variable size grants on a regular basis
  - Offers unicast polls on a regular basis and CMs are allowed to use contention opportunities and piggybacking
- Best Effort (BE) Service
  - Provides efficient service to Best Effort Traffic
  - Uses contention and piggybacking for requests
  - Limited QoS Support

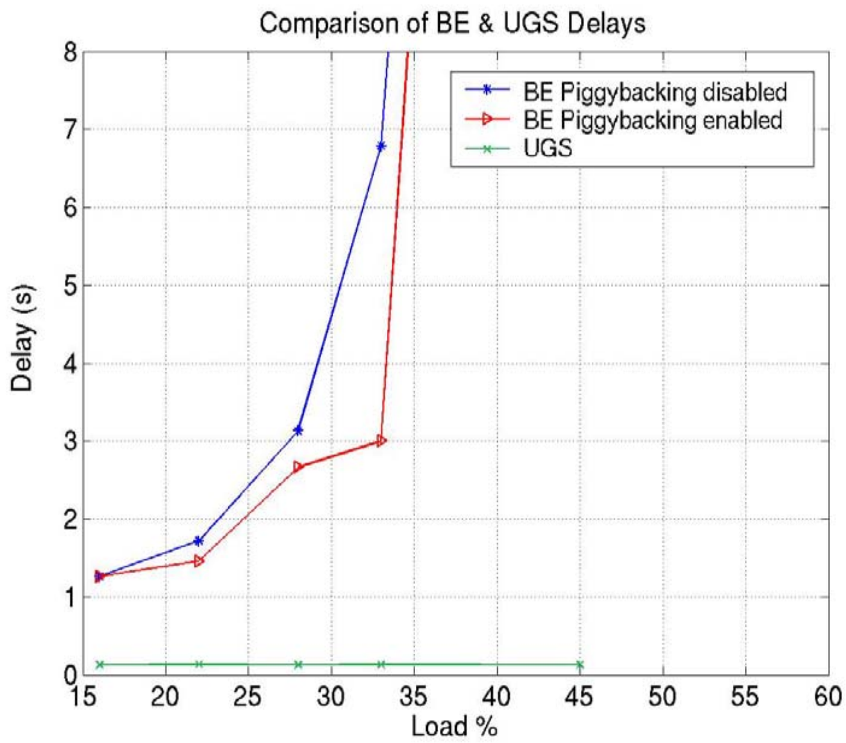
Performance Evaluation of DOCSIS 1.1 QoS

- Key performance attribute studied - Mean Access Delay
- Simulations were done in OPNET
- Comparison of BE and UGS performance
- Effect of DOCSIS 1.1 QoS features on performance
  - Fragmentation - sending a portion of packet frame during a reserved slot time
  - Concatenation - sending more than a frame during a transmission opportunity
  - Piggybacking - a request carried in the next outgoing data frame
  - Traffic Priority - CMTS uses Traffic Priority attribute for determining precedence in grant generation

Experiment 1: Comparison of Best Effort & UGS Delays

- UGS
  - UGS flows are allowed to reserve certain portion of the bandwidth
  - No transmission requests are needed; hence low, constant access delay
- BE
  - “request grant,request grant” pattern
  - Has to contend for sending requests
  - May result in collision and thus increased delay due to retransmissions
  - Piggybacking: requests are piggybacked to outgoing data and thus delay is reduced

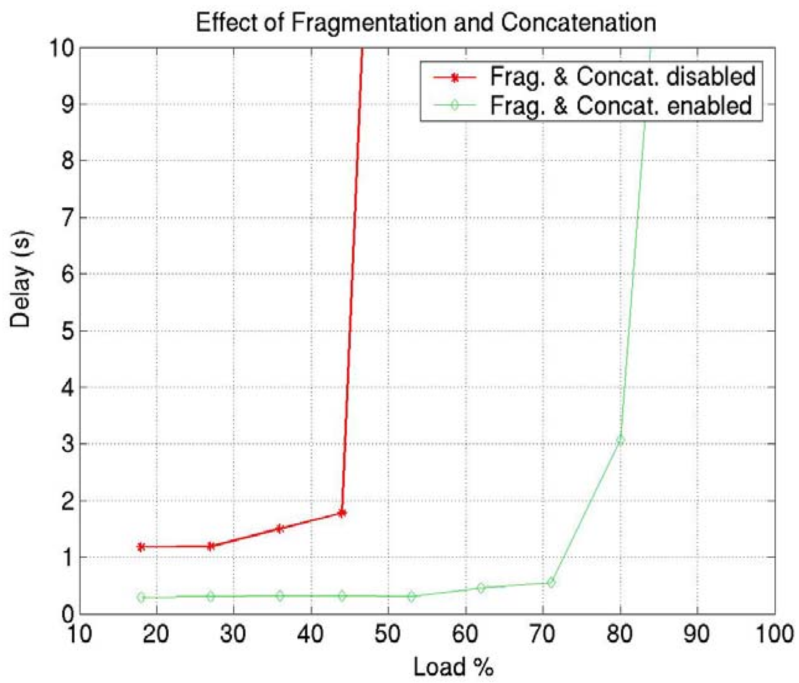
Experiment 1: Comparison of Best Effort & UGS Delays (*cont.*)



- Load increased by adding BE stations
- Fragmentation & Concatenation were disabled
- Exponential packet lengths & inter-arrival times
- Backoff start = 7

Experiment 2: Effect of Fragmentation & Concatenation

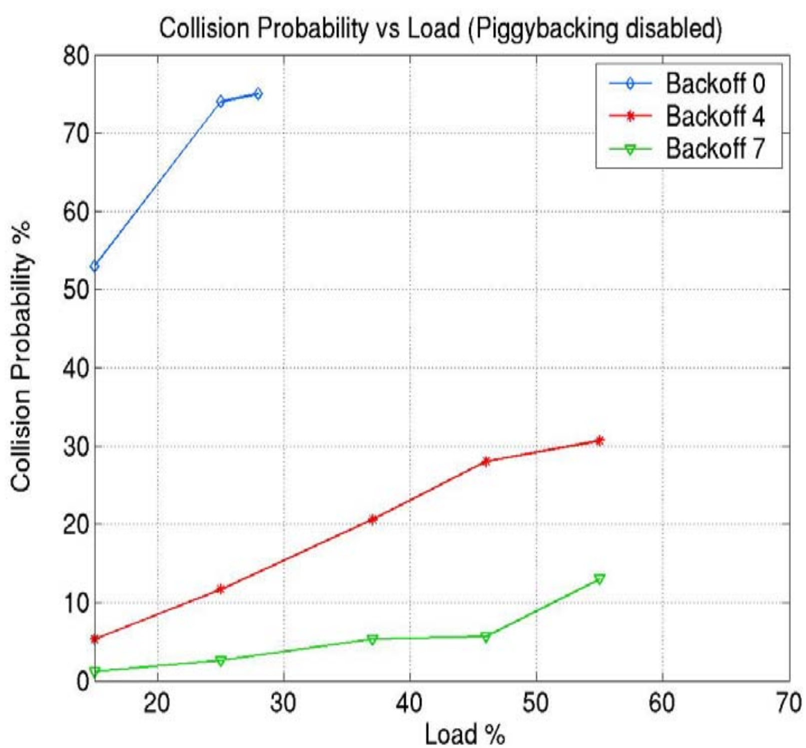
- Load was increased by adding BE stations
- Exponential Packet Lengths and Exponential Inter-Arrival Times
- Fragmentation & Concatenation improve performance considerably
- Piggybacking was enabled;Backoff start =7



Contention Resolution Algorithm (CRA) Overview

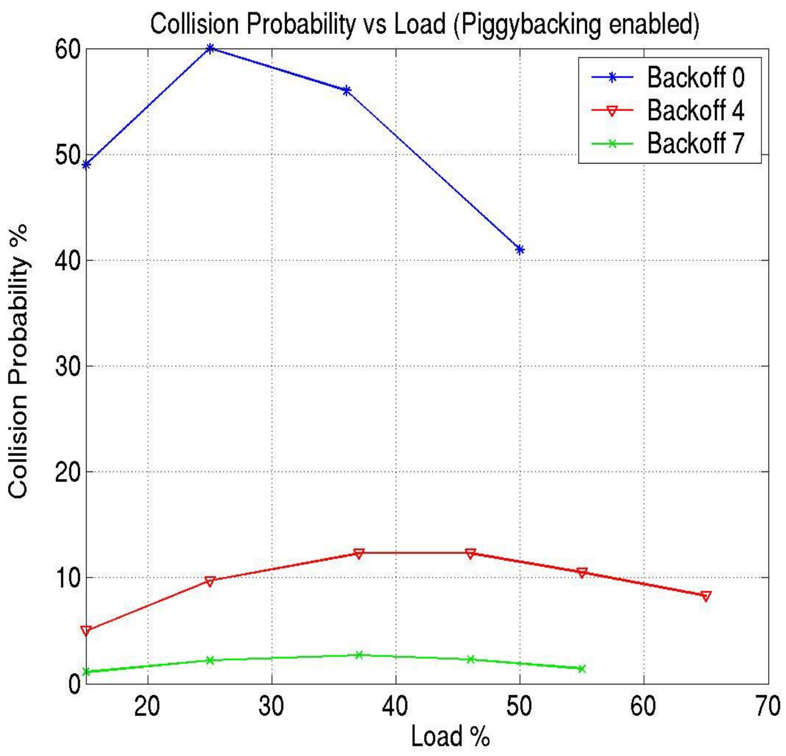
- CRA triggered by request collision
- Supported CRA: Truncated Binary Exponential Back-off
- Specified by an initial backoff window (Back-off start) and a maximum back-off (Back-off End)
- CM randomly selects a value within its back-off window - $[0,2^{\text{Backoff}}]$
- Random number indicates the number of contention opportunities the CM must defer before transmitting requests
- On collision , increases the back-off window by a factor of 2 (less than back-off end) and repeats deferring process

Experiment 3: Effect of Backoff Start on Collision (piggybacking disabled)



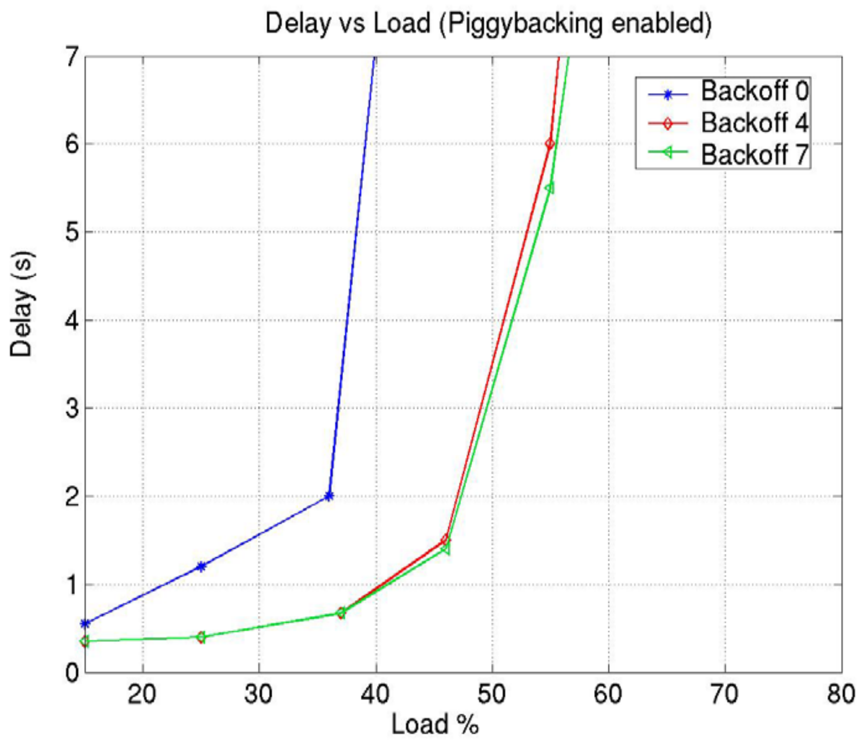
- Packet lengths and inter-arrival times were made constant
- Since Piggybacking is disabled, collision probability increases with load
- As backoff start value increases, collision probability decreases

Experiment 3: Effect of Backoff Start on Collision (piggybacking enabled)



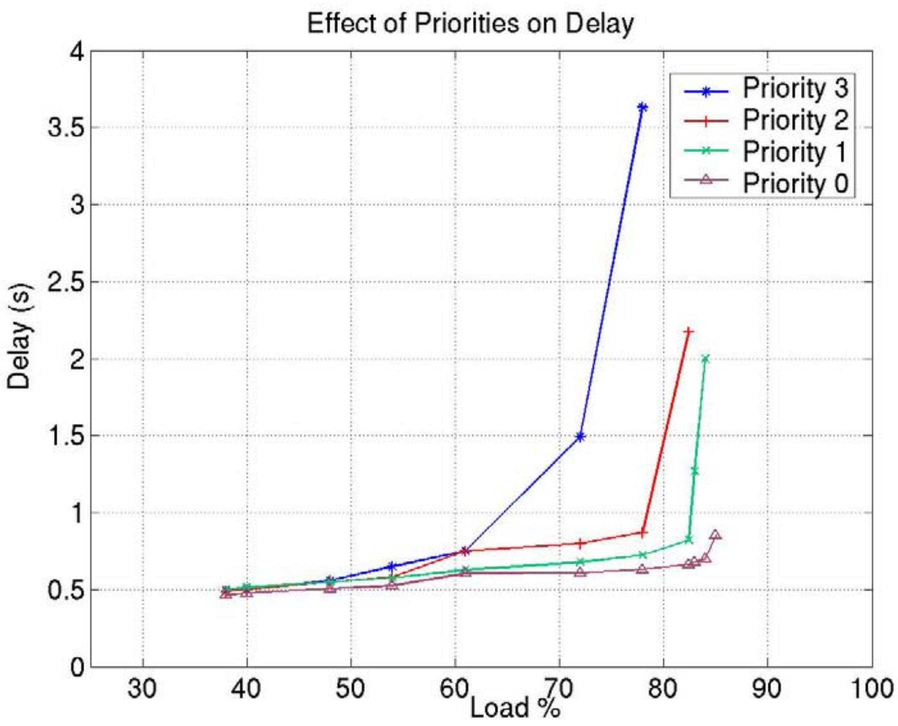
- Less Collision probability compared to the piggybacking disabled case since frequent contention is not necessary
- As load increases, collision probability decreases since contention load decreases

Experiment 3: Effect of Backoff Start on Delay



- Backoff 0 suffers maximum delay
- Backoff 4 produces a marked improvement in performance
- Backoff 7: reduced collision is offset by longer backoff delay

Experiment 4: Effect of Traffic Priorities on Delay



- CMTS uses Traffic Priority attribute for determining precedence in grant generation
- Priorities do not affect contention
- Traffic Priorities range from 0-7 applicable to BE, rtPS & nrtPS, 0 being the highest
- Hence high priority stations have lower access delays

Summary of Results

- Performance Evaluation of HF^2Q
  - HF^2Q behaves as FIFO when the total system load is less than 1
  - Behaves as WFQ when the system is heavily overloaded
  - Throughput behavior of HF^2Q confirms these properties
- DOCSIS 1.1
  - UGS always receives bounded delay characteristics
  - Fragmentation and Concatenation improve performance when system is overloaded
  - Piggybacking improves delay characteristics
  - CRA plays an important role in BE performance
  - Traffic priority has significant effect on BE performance

Contributions & Future Work

- Performance characterization of a new queuing discipline - HF^2Q was done
- Performance evaluation of DOCSIS QoS
- Scope for future work with regard to HF^2Q
  - Analytic explanation as to why HF^2Q behaves as FIFO in an underloaded case
  - Finding a closed form expression for the mean waiting time for WFQ
- Performance evaluation studies of two other traffic classes that DOCSIS 1.1 mentions namely rtPS and nrtPS yet to be done

Discussion

- Thank you!