



Hyperchip Inc.

AQUA

For Assured QoS in the ISP Network

Dongli Zhang

1	Introduction	3
1.1	Background.....	3
1.2	Solutions.....	3
1.3	Challenges	4
2	Architecture of the Scheme	5
3	AQUA	6
3.1	Architecture for AQUA Implementation.....	6
3.2	AQUA’s Objectives and Philosophy	7
3.3	Inputs to and outputs from AQUA	7
3.4	The Basic AQUA Algorithm for Computing DP and DProb	8
3.5	AQUA Protocol and packet	10
4	References	14

1 Introduction

1.1 Background

ISPs are upgrading their networks for faster Internet access to support high-bandwidth consumer applications such as P2P, IPTV, Web 2.0 and online gaming.

For example, P2P technology such as BitTorrent distributes large data files by dividing them into small pieces and sending them through multiple sources; after all the data is received, the file is reassembled as a whole. While this method of file sharing is much faster and more efficient than relying upon one centralized server, it causes traffic-management problems for ISPs because P2P protocols currently download large chunks of data from sources wherever they can be found, without regard to network efficiency. The rapid growth of high-bandwidth-consuming applications makes it very difficult for ISPs to determine just how much to invest in building out capacity.

The fundamental problem is that all network resources are limited, and the bandwidth capacity is easily filled by the applications. The purpose of this document is to describe an architecture and algorithms for Bandwidth Control and congestion avoidance in ISP networks.

1.2 Solutions

The most difficult area of class-of-service is congestion avoidance. While today's routers depend solely on WRED and TCP back off for this function, Hyperchip has developed a suite of internal traffic management algorithms that complement WRED.

Two attributes minimize the effectiveness of WRED-only implementations. First, WRED determines congestion by looking at average buffer utilization, which detects congestion only after it has built up a queue. Second, WRED drops packets at the congestion point, instead of preventing a congestion point from forming; packets doomed to be discarded consume resource on the way to the congestion point, and packets from one source can even block higher-priority packets from other sources from reaching the congestion point.

To overcome the shortcomings of WRED, Hyperchip developed AQUA™ – Adaptive Queue Utilization Algorithm. AQUA utilizes the drop probability curves configured for WRED and the bandwidth configurations for each queue. In order to detect incipient congestion, AQUA calculates the average incoming bandwidth for each queue/drop precedence combination per port. This provides a proactive determination of congestion versus WRED's reactive determination. This also allows AQUA to detect administrative congestion as well; AQUA will detect bandwidth utilization that exceeds the configured maximum bandwidth per queue and can make appropriate drop probability calculations.

While these features alone would be an improvement, Hyperchip extends AQUA's effectiveness. Instead of the egress making the dropping decision, the drop probabilities are broadcasted to all ingresses. The ingresses then perform the dropping decision for each destination based on the egress conditions. This ensures minimum latency for packets that can be transmitted, regardless of congestion severity. Fairness among flows is also guaranteed since each ingress port receives the identical drop probability. This distribution thus guarantees an equal proportional dropping policy for all ingresses.

This current project extends the AQUA from the router into the ISP network. Its mission is to take care of long-term congestion relief and ensure that a global precedence-based dropping is respected, thus enabling Internet Service Providers to deliver uncompromised differentiated services to both their business and residential customers.

- | AQUA™ | WRED |
|---|--|
| <ul style="list-style-type: none">• Limits bandwidth on switch fabric• Applied on ingress depending upon bandwidth utilization of egress• Distributed throughout entire router• Based on average bandwidth utilization | <ul style="list-style-type: none">• Limits size of buffers• Applied on egress depending upon queue congestion• Local drop probability based on egress buffers• Based on average queue depth |

1.3 Challenges

If the AQUA is extended into the network, it will be not an internal protocol within a single router, and will thus raise network management issues:

- Does the ISP really want to deploy the QoS mechanism in their network?
- Is this necessary to avoid the congestion in the ISP network?

Today AQUA works by measuring at a point of overspeed before a bottleneck. But in the proposed application, most of the data is flowing from the provider to the end-point, rather than from the end-point to the provider. Since the subscriber's household network usually has more bandwidth than the link coming in does, the end-point is not a point of overspeed before a bottleneck; it is a point of overspeed after a bottleneck.

One could consider having AQUA manage the outgoing bandwidth, where at least the end-point sits before the bottleneck. But the AQUA-enabled device wouldn't be shoveling the bits, so how would it control the queues? And to whom would it send the slow-down messages – the subscriber's applications? The best that AQUA could do for

outgoing bandwidth would thus be to provide finer control over which queues were allowed to send, and even for that it would have to control the end-point's own QoS mechanism. Since the end-point probably already prioritizes VOIP over web and web over P2P, there is probably little value to adding for outgoing bandwidth.

2 Architecture of the Scheme

When AQUA is running on a CPU, it will produce a generalizable QoS appliance. Simply feed AQUA the bandwidth allocations and/or priorities for configuration, and then feed it the ongoing bandwidth statistics, and AQUA will figure the drop probabilities that will tune the traffic so that the right traffic goes through and all traffic that goes through receives great service. The AQUA computation engine remains independent of how the measurements are gathered and how the queues are shaped, although some end points will be able to supply more information than others.

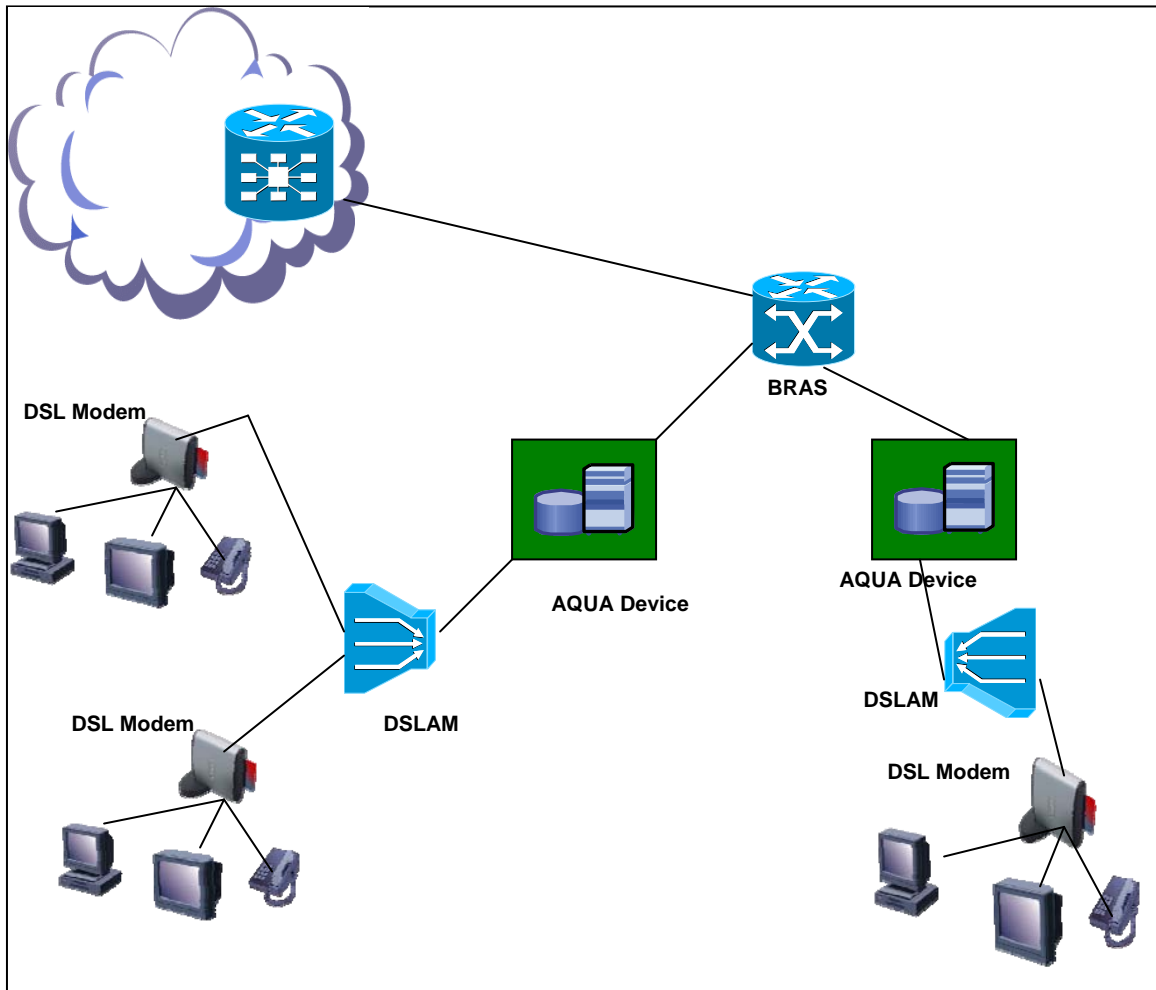


Figure 1: Architecture of the project

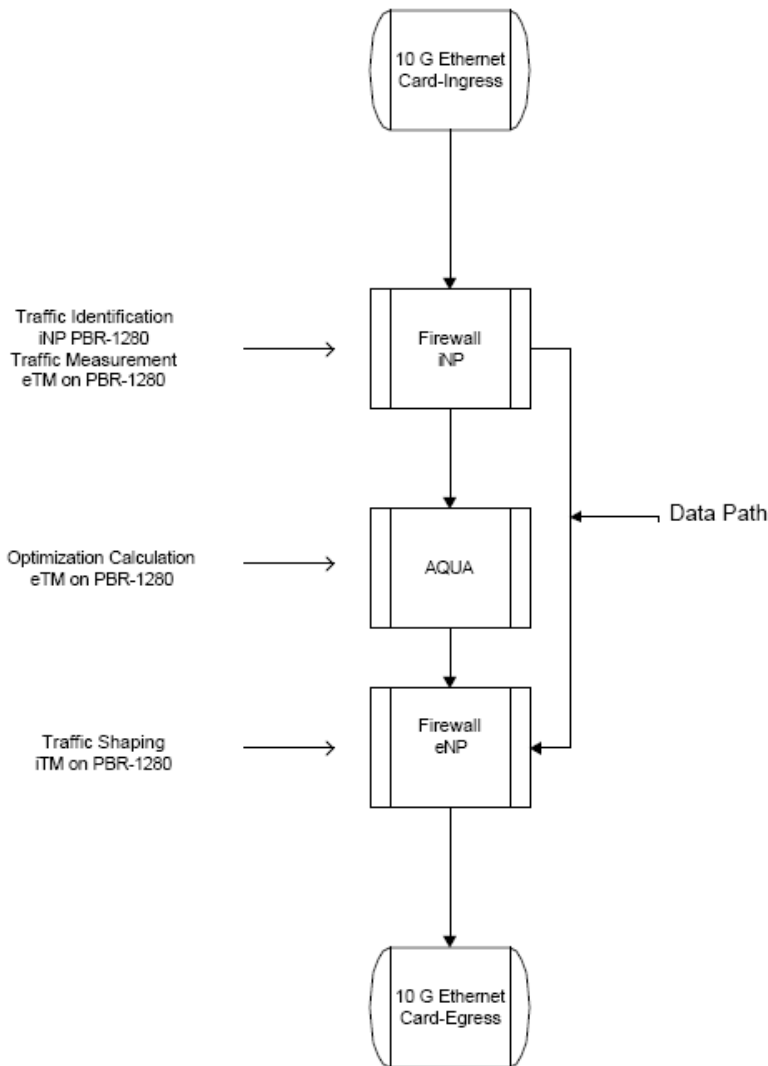
The BRAS will terminate the user's PPP session. The QoS policy on the BRAS allows per PPP session policing and shaping according to the SLA. Once the traffic goes out of BRAS, there is no congestion on the BRAS to DSLAM link. From DSLAM to user modem, congestion does not exist either.

In the upload direction, the policy is also applied on the BRAS. There is no congestion from DSLAM to BRAS.

3 AQUA

This section describes AQUA's objectives, its philosophy, configuration parameters, and algorithm implementation.

3.1 Architecture for AQUA Implementation



3.2 AQUA's Objectives and Philosophy

The main objectives of AQUA are:

- Long-term congestion relief, through dropping excess packets based on their drop-precedence
- Ensuring that a global (or system-wide) precedence-based dropping is respected.

3.3 Inputs to and outputs from AQUA

In order to function, AQUA needs to have queue information available for its calculations, and to have a way of controlling queues.

The simplest way to provide these is the way it is done in the PBR-1280 – all of the packets flow through AQUA-controlled queues and through the AQUA measurement point. However this requires a network processor to classify the packets and fast memory to queue the packets, and it is questionable how much bandwidth even a modern processor could accomplish this for without dedicated custom hardware.

Devices such as firewalls already have network processors that are classifying the packets that pass through them, and also manage queues and can drop packets based on that classification and configuration parameters. Firewalls also have CPUs on which AQUA code could run, obtaining information from the firewall, computing the bandwidth shaping required to prevent down-stream congestion, and dynamically reconfiguring the firewall's parameters to implement that shaping.

Today AQUA runs in the PBR's traffic managers to shape bandwidth for ports that belong to the PBR's egress network processor. In an analogous way, AQUA on a firewall could shape the bandwidth for the multiple lower-speed interfaces on a subsequent access box, such as a DSLAM or a PON OLT. AQUA would thus keep the access box from receiving more bandwidth for one port than that port could handle, which could crowd out data for other ports. This would protect an access box that has weak traffic management that cannot handle the line rate of its input.

To the extent that AQUA is implemented within a single firewall and the downstream box already has decent traffic management and a single interface as fast as the firewall's input, then such a simple AQUA implementation can only add its superior algorithm. For example, AQUA can add administrative congestion management, limiting bandwidth for various users to their configured limits unless there is extra bandwidth, which it would share fairly.

When AQUA is not limited to a single box, AQUA can add additional value. AQUA has a congestion information distribution mechanism, and so multiple boxes running AQUA can share information. For example, if an access box has multiple input that pass through AQUA boxes (such as a local content feed and an internet feed), then AQUA can coordinate between them to make sure that in aggregate they don't overload the access box's traffic management.

The AQUA measurement function should be before the most restrictive congestion point, with how far before being a matter of convenience. If the measurement point is far enough upstream that there are still multiple paths to reach that congestion point,

then each path needs an AQUA measurement function, and the measurements on the various paths for a given point must be consolidated for the shaping calculations to be made.

The farther upstream that the shaping function of AQUA occurs, the more value AQUA can add by protecting downstream resources. If AQUA's shaping function is implemented where there are multiple paths to reach the congestion point, then each path needs an AQUA shaping function and the results of the shaping calculation have to be distribute to these shaping functions.

3.4 The Basic AQUA Algorithm for Computing DP and DProb

Four quantities that will be used to as inputs to compute the AQUA DP and DProb for each queue. These are:

- `current_number_of_AF (q)`
- `average_number_of_AF (q)`
- `average2_bandwidth_allocated (q)`
- `average2_bandwidth_received (q)`

Another quantity, which is not explicitly shown in Figure 1, is:

- `ratio_of_aqua_to_share`, which is the ratio of the AQUA period to the SHARE period (i.e. `aqua_period / share_period`)

Figure 2 shows the pseudo-code that describes the implementation of AQUA, and which produces AQUA's Global Dropping Probabililty (GDP). Note that DP and DProb can be straightforwardly derived from the GDP.

```

EVERY aqua_period DO the following for each Queue (q) within each Interface:
{
  IF (current_number_of_AF(q) > 0)
  {
    IF ((average_number_of_AF(q) / ratio_of_aqua_to_share) > 0.75)
    {
      GDP(q) = GDP(q) + 8;
    }
    ELSEIF ((average_number_of_AF(q) / ratio_of_aqua_to_share) > 0.50)
    {
      GDP(q) = GDP(q) + 4;
    }
    ELSEIF ((average_number_of_AF(q) / ratio_of_aqua_to_share) > 0.25)
    {
      GDP(q) = GDP(q) + 2;
    }
    ELSEIF ((average_number_of_AF(q) / ratio_of_aqua_to_share) > 0.125)
    {
      GDP(q) = GDP(q) + 1;
    }
    ELSE
    {
      GDP(q) = GDP(q) + 1;
    }
  }
  ELSE
  {
    IF ((average2_bw_received(q) / average2_bw_allocated(q)) < 0.75)
    {
      GDP(q) = GDP(q) - 16;
    }
    ELSEIF ((average2_bw_received(q) / average2_bw_allocated(q)) < 0.875)
    {
      GDP(q) = GDP(q) - 8;
    }
    ELSEIF ((average2_bw_received(q) / average2_bw_allocated(q)) < 0.9375)
    {
      GDP(q) = GDP(q) - 4;
    }
    ELSEIF ((average2_bw_received(q) / average2_bw_allocated(q)) < 0.96875)
    {
      GDP(q) = GDP(q) - 2;
    }
    ELSEIF ((average2_bw_received(q) / average2_bw_allocated(q)) < 0.984375)
    {
      GDP(q) = GDP(q) - 1;
    }
    ELSE { // Do nothing; this should be the convergence region
    }
  }
  IF (GDP(q) < 0)
  {
    GDP(q) = 0;
  }
  ELSE IF (GDP(q) > 1023)
  {
    GDP(q) = 1023;
  }
}

```

Figure 2: Pseudo-code of the basic AQUA algorithm used for computing the GDP of each (Interface, Queue) combination

3.5 AQUA Protocol and packet

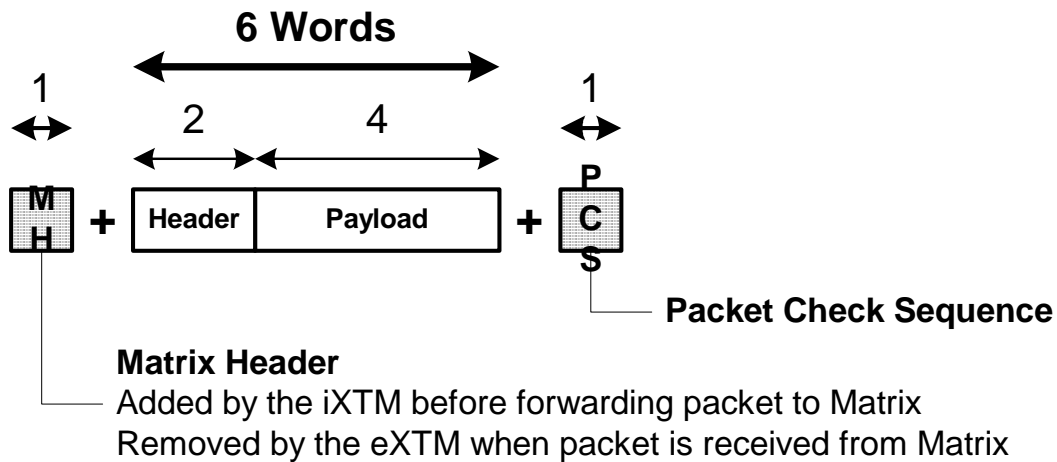
AQUA packets are used only to support the AQUA protocol. For a given queue these packets are always multicast from one AQUA calculation function to all AQUA shaping functions that can send packets for that queue. In the PBR these are forwarded in the same way as Multicast Packets, and there are 16 multicast addresses reserved for AQUA packets. They are used to support failure and load distribution in the switch fabric.

In a single-firewall implementation, the AQUA measurement, calculation and shaping are all in the same box, so “multicasting to all” could be simply writing to memory.

3.5.1 AQUA Packet Format

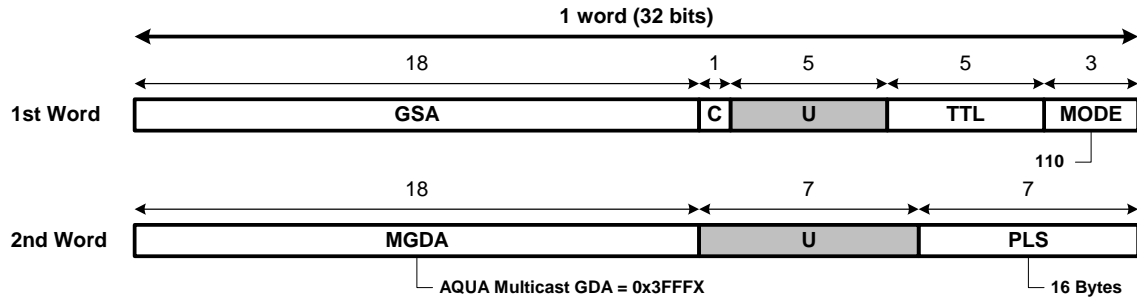
In the PBR-1280, an AQUA Packet is built of a 2-word AQUA Header and a 4-word Payload. A Packet Check Sequence word is appended by the chipset at the end of each packet for error detection. Please refer to the “Protection” section of this document for more details. The xTMs are responsible for handling the Matrix Headers, which are 1-word long.

This format is customized for the PBR. Of the matrix header and AQUA Packet header, the only AQUA-specific fields are the GSA (Queue identifier) and four bits MGDA for the multicast group to use. The GSA and the AQUA payload are the parts that the AQUA shaping engines need; the rest is simply for transport across the PBR fabric.



3.5.2 AQUA Packet Header

Field Distribution:



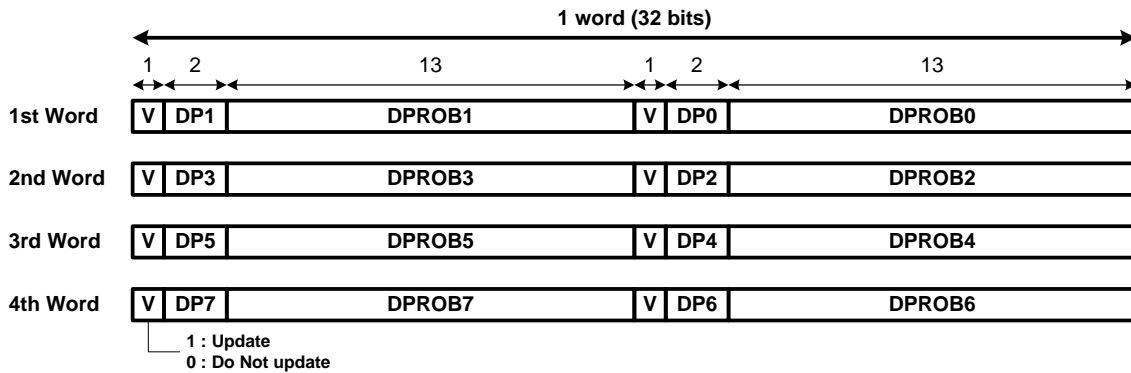
Field Description:

Name	Position	Description
GSA	Word 1, 31:14	Global Source Address.
C	Word 1, 13	CPU Bit. When asserted, this bit indicates that the AQUA Packet relates to a HLC or HCC Processor address. When de-asserted, this bit indicates that the AQUA Packet relates to a Data Port.
U	Word 1, 12:8	Unused.
TTL	Word 1, 7:3	Time To Live.
MODE	Word 1, 2:0	Addressing Mode. Always equal to 110.
MGDA	Word 2, 31:14	AQUA Reserved Multicast Global Destination Address range (0x3FFF0 to 0x3FFFF)
U	Word 2, 13:7	Unused.
PLS	Word 2, 6:0	Payload Size. Always equal to 16 Bytes.

3.5.3 AQUA Packet Payload

The Payload of an AQUA Packet contains the information necessary to maintain the AQUA protocol. Each half-word of an AQUA Payload word is related to a queue. Each AQUA Packet has information of up to 8 queues. When the C field of the AQUA Packet header is equal to 1, only the first word of the payload contains valid information, i.e., the V fields on words 2, 3 and 4 will be always de-asserted.

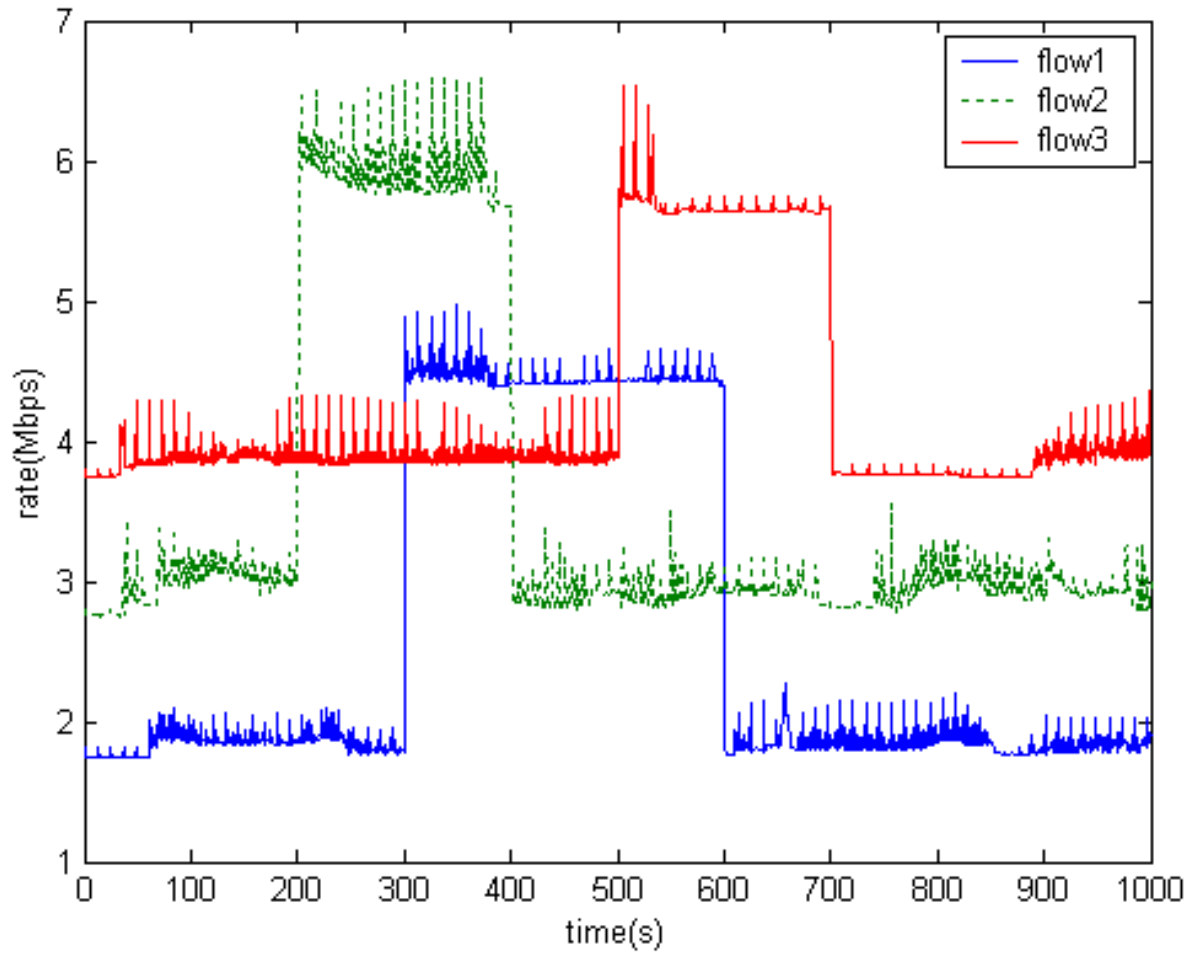
Field Distribution:



Field Description:

Name	Position	Description
V (0:7)	Words 1/2/3/4, Bits 31 and 15	Valid Update. When set, indicates that its corresponding status update (DP + DPROB) is valid.
DP (0:7)	Words 1/2/3/4, Bits 30:29 and 14:13	Discard Precedence [4].
DPROB (0:7)	Words 1/2/3/4, Bits 28:16 and 12:0	Discard Probability [4].

3.5.4 Some Simulation Results



4 References

1. [TCP Tunnels: Avoiding Congestion Collapse \(2000\)](#)
2. [Van Jacobson, Michael J. Karels. Congestion Avoidance and Control \(1988\).](#) *Proceedings of the Sigcomm '88 Symposium*, vol.18(4): pp.314–329. Stanford, CA. August, 1988. This paper originated many of the congestion avoidance algorithms used in TCP/IP.
3. [RFC 2001](#) - TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms
4. [RFC 2581](#) - TCP Congestion Control
5. [RFC 3390](#) - TCP Increasing TCP's Initial WindowOriginal RFC3390]
6. [TCP Congestion Avoidance Explained via a Sequence Diagram](#)
7. [RFC 2309](#) - April 1998: Recommendations on Queue Management and Congestion Avoidance in the Internet
8. [Sally Floyd: RED \(Random Early Detection\) Queue Management](#)
9. Sally Floyd, Van Jacobson. [Random Early Detection Gateways for Congestion Avoidance](#) (1993). *IEEE/ACM Transactions on Networking*, vol.1(4): pp.397–413. Invented Random Early Detection (RED) gateways.
10. [An Analytical RED Function Design Guaranteeing Stable System Behavior](#)
11. [RFC 3168](#) - The Addition of Explicit Congestion Notification (ECN) to IP
12. [Comparative study of RED, ECN and TCP Rate Control \(1999\)](#)
13. [Active Queue Management](#)
14. [Enabling Dynamic Buffer Limiting](#)
15. "Deploying IP and MPLS QoS for Multiservice Networks: Theory and Practice" by John Evans, Clarence Filstis (Morgan Kaufmann, 2007, [ISBN 0-12-370549-5](#))