Strategien zur Paketverarbeitung bei Dienstgüte-Unterstützung

Krisna Haryantho
Betreuer: Heiko Niedermayer
Seminar Future Internet SS2011
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: haryanth@in.tum.de

ABSTRACT
Computer networks today often have limited resource. This limitation is one of the reasons why sometimes packets sent over the network are delayed or even dropped. The simple best-effort network or service does not support quality of service [from Wiki]. This paper introduces several bandwidth management techniques and strategies that can be used to achieve quality of service over a best effort network.

Keywords
DiffServ, Quality of Service, Traffic Shaping, Traffic Policing, RED, WRED, Two Rate Three Color Marker, Leaky Bucket, Token Bucket

1. INTRODUCTION
In computer networks, the presence of bottleneck links is hard to prevent. In the simplest operation mode (no shaping, limiting, policing, etc. used), packets that are sent through a network will be delivered in a best effort manner. This means when a packet arrives at a network point, it will either be forwarded immediately when there is enough bandwidth to do so, or discarded otherwise. In the case of aggressive hosts, where hosts try to send packets using their full bandwidth, this would lead to traffic congestion at bottleneck links. Some transport layer protocols such as TCP provide a congestion control mechanism. TCP starts by sending packets at a slow speed, and gradually increasing this speed as far as possible. When it detects congestion (usually signalized by dropped packets), it will reduce the transmission speed. When all packets can be delivered successfully, TCP will try to increase the speed again. This process continues until TCP 'finds' the ideal transmission rate for the current transfer. This mechanism is called the TCP slow-start.

The best-effort network cannot provide Quality of Service due to its somewhat indeterministic behaviors. This means, there is no way of guarantying that packet are received, and this within a specific delay and jitter (standard deviation of the delay). Currently the most common transport protocols in the packet-switched network are the TCP and UDP. While TCP provides a congestion control mechanism, UDP does not. Given this fact, UDP (or in general non-responsive flows) will tend to dominate the available bandwidth, starving out the remaining TCP traffic (or in general the responsive flows) [1].

While increasing capacity by expanding the network can address the traffic congestion issue, it does not scale up and is in most cases not the most cost-efficient way of dealing with this problem. This is where the term Quality of Service comes into play. Quality of service (QoS) is the ability to provide different priority to applications, users, data flows in order to guarantee a certain level of performance (Citation needed!) – In the best effort setting, all packet have the same priority and are treated the same way. This performance includes transfer rate, latency, jitter and drop probability. Performance is an important issue for real-time applications such as multimedia streaming, video conferencing, or online gaming, as they often require a steady transfer rate and are delay sensitive.

In the following sections we will discuss two bandwidth management mechanisms that can be used to provide quality of service. Section 3 talks about traffic shaping and algorithms that are used to realize it. Section 4 talks about congestion mechanism and some of the most popular methods of to avoid congestion. Section 5 provides a short introduction to DiffServ, the current accepted way of implementing quality of service. Section 6 concludes the paper with some closing remarks on the discussed topics.

2. TRAFFIC SHAPING/RATE LIMITING
Network providers and their customer agree upon a certain customized traffic profile - this is called the Service Level Agreement (SLA). To keep their network running smoothly, providers would want to ensure that the customers are not generating more traffic than what is specified in the SLA. One way to control the bandwidth is by doing traffic shaping. Traffic shaping is a strategy to increase performance on a resource-limited network. It works by delaying some or all of the packets in a traffic stream in order to bring the stream in compliance with a traffic profile [2].

Principally each incoming packet will be verified against a certain traffic policy. Packets will be forwarded only if it conforms to the policy. Otherwise it will either be dropped, delayed, or forwarded with lower priority. There are two basic algorithms that are used for implementing traffic shaping – the token bucket algorithm and the leaky bucket algorithm. These two algorithms will be discussed in the following subsection. But before that, let us consider the two-rate three color marker as a method to classify/differentiate traffic.

doi: 10.2313/NET-2011-07-2_04

2.1 Two Rate Three Color Marker

The two rate three color marker provides a way to classify IP packets. The two rates being used are called CIR und PIR.

Committed Information Rate (CIR) is the data rate that the service provider is guaranteeing to its subscriber. A service level agreement usually demands that all traffic at the rate of CIR or less will be delivered to its destination with high probability [3]. If a network is so provisioned, that it is capable of carrying the sum of all subscriber CIRs, that network would be underutilized. This is because it is statistically unlikely that all subscribers generate traffic at CIR at the same time. Because it is not cost efficient to operate an underutilized network, the provider would often allow the subscriber to generate traffic at a higher rate than CIR. This rate is called the *Peak Information Rate* (PIR).

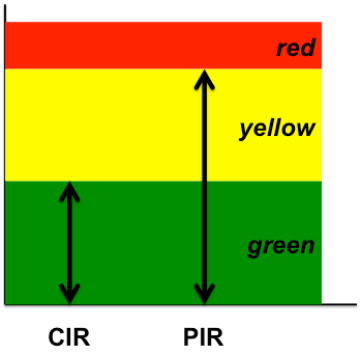

Figure 1: Different coloring for each rate

A Two Rate Three Color Marker meters IP packet stream and marks its packets green, yellow, or red [4]. The color is coded in the DiffServ field of the IP header. A packet is marked green if the stream is at or below the CIR, yellow if it is higher than the CIR but lower or equal the PIR, and red otherwise (see Figure 1: Different coloring for each rateFigure 1).

A very common practice is to forward green packets and assure their delivery, forward yellow packets with best effort. While red packets will usually be dropped, some systems might still forward red packets as if they were yellow packets [3].

2.2 Token Bucket

The token bucket algorithm can be understood as a container (a bucket) that is continuously filled with tokens at a certain rate [3]. This bucket has a size, which is the number of tokens it can hold. When the bucket becomes full, no more tokens can be added to it – any new tokens added will simply be dropped.

This token can be thought of as a stamp for each packet that needs to be forwarded. The algorithm works then as follows. Whenever a packet arrives, the algorithm will try to remove a certain amount of tokens from the bucket. Usually we measure token in the unit of byte with one token conforming to one byte. Thus packet with 500 bytes size requires 500 tokens.

By using token bucket, one can get a controlled (‘shaped’) output traffic from any input traffic. The rate of the desired output is regulated by the token fill rate. Token bucket also allows the presence of short duration burst traffic. The maximum allowed burst traffic conforms to the bucket size. Figure 2 illustrates the token bucket algorithm. Here the unregulated input traffic are shaped into a regulated output traffic with a constant rate.

Now consider the dual-rate token bucket algorithm, the extension of this algorithm where there are two buckets being used. The first bucket has the size X and is filled with the rate of the CIR (The CIR bucket). The second bucket has the size Y and is filled with the rate of the PIR (The PIR bucket). Both buckets start full and the algorithm works as following (See Figure 3): whenever a packet arrives, it checks the PIR bucket whether it currently holds enough tokens to forward the packet with. If this is not the case, the packet is marked as not conforming (red) and can later be dropped. Otherwise the algorithm will remove tokens from the PIR bucket and checks the CIR bucket whether it also holds enough tokens. If this is the case, the algorithm will remove tokens from CIR bucket and the packet will be marked as conforming (green) and will be forwarded. Otherwise the packet will be marked as exceeding (yellow) and will be forwarded in the best effort manner.

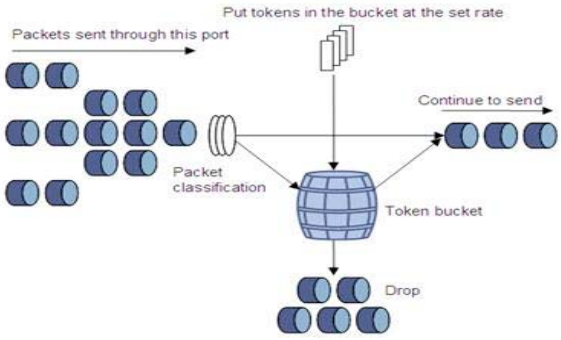

Figure 2: Token bucket algorithm (Source www.h3c.com/…/200701/195599_57_0.htm, accessed on 20th April 2011)

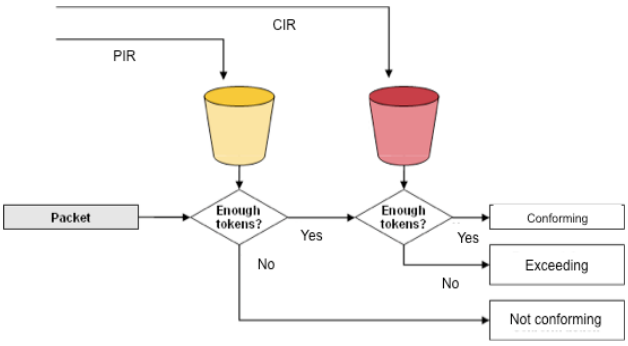

Figure 3: Dual-rate token bucket algorithm

Assume CIR is 1500 KB/s and PIR 3000 KB/s. Both buckets are set to 1500 Bytes. When a 1500 Byte packet arrives, 1500 Bytes will be removed from both bucket. The bucket is empty and that packet will be green. Then a second 1500 Byte packet arrive ½ millisecond later. By this time 1500 Bytes (½ ms x 3000 KB/s) of token has been added to the PIR bucket. The CIR bucket has 750 Bytes (½ ms x 1500 KB/s) of token. This second packet will be yellow, and 1500 Bytes are removed from the PIR bucket only. If a third 1500 Byte packet arrives ½ millisecond later, both bucket will contain 1500 Bytes and that packet will be green.

doi: 10.2313/NET-2011-07-2_04

2.3 Leaky Bucket

The leaky bucket algorithm is the other often used algorithm to shape traffic. While token bucket is typically implemented for IP networks, leaky bucket is usually implemented for ATM networks. In ATM the term cell is used instead of packet. And analog to the IP network, in ATM network the Sustained Cell Rate (SCR) and the Peak Cell Rate (PCR) are used in a similar manner to CIR and PIR respectively.

The algorithm is similar to the token bucket, only in the leaky bucket case the packets are filled into the bucket. The name leaky bucket comes from an analogy of a bucket that has a hole at the bottom. Water (ATM cell) can be filled into the bucket at any rate and leaks through the bottom hole at a certain rate until the bucket becomes empty. If the bucket is full, any added cells will be discarded.

Whenever an ATM cell arrives, the algorithm checks whether there is enough space in the bucket to contain every byte in the cell. If there is enough space, the packet is added to the bucket, otherwise it will be discarded. The desired output rate is thus the leak rate of the bucket. The leaky bucket algorithm allows input burst, meaning it will take packets at any rate as long as there is still enough space in the bucket left. The maximum burst size is thus the bucket size.

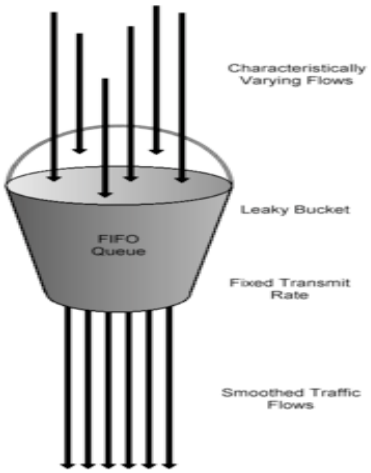

Figure 4: Leaky bucket used as queue to shape traffic (Source: Wikipedia)

Similar to the token bucket, there is a variation of the leaky bucket where two buckets are used. The first bucket is used for the guaranteed traffic. It leaks at the SCR. The second bucket is used for the excess traffic that does not fit in the first bucket anymore. This bucket leaks at PCR. The algorithm will try to fill all incoming traffic to the SCR bucket. If this bucket becomes full, the excess traffic goes to the PCR bucket and will be forwarded in a best effort basis (similar manner to yellow marked IP packet).

3. CONGESTION AVOIDANCE

The core network consists of switches and routers. These devices are prone to congestion [5]. Congestion occurs when the aggregate traffic coming through the *ingress* interface (incoming traffic) has a higher rate than that which can be successfully processed by the *egress* interface (outgoing traffic). Congestion can also occur due to the inability of the switch/router CPU to handle the size of the forwarding table [5].

In the default setting, when the internal queue of an interface is full, any new incoming packets will be dropped. Due to the first-come first-served nature of the queue, overflowed packets are dropped without regarding their previous classification or marking. This phenomenon is called *tail dropping*.

As mentioned previously in section 1, TCP provides a mechanism to control congestion in the network. With regard to the tail drop phenomenon, there are two major problems that can arise when using TCP congestion control mechanism:

- In an aggregated traffic, a large number of TCP packets that are dropped (as a result of tail dropping) can cause a large number of hosts to detect congestion. As a counter measure they will immediately reduce their transmission rate. This will make the corresponding link underutilized for a short period until the TCP hosts speed up their transmission rate again. The bandwidth adjustment can happen over and over again at the same time across all active TCP connections. This is known as the *TCP Global Synchronization*.
- In real networks there is also non-responsive traffic. Non-responsive traffic, such as the UDP, tends to be more aggressive in using available bandwidth and -in the case of UDP- lacks of congestion control mechanism. By the time a link becomes underutilized (due to result of TCP global synchronization), it will consume the remaining bandwidth leaving no resource for TCP. This effect is called *TCP starvation*.

Due to these issues, the traditional queue mechanism is not enough to guarantee Quality of Service. To do this, switches or routers must be equipped with a mechanism to queue and service high priority traffic before lower priority traffic [5]. Furthermore it must also be possible to drop lower priority packets before higher priority packets during periods of congestion.

To counter the tail drop effect resulting from the use of traditional queue, the term *active queue management* (AQM) was introduced. AQM takes advantage of the congestion control mechanism provided by TCP. Rather than dropping packet after the queue is full, it prevents the queue (an *active queue*) to become full. Commonly there are two ways to do this. The first is done by dropping packets, the second is by ECN-marking packets.

3.1 Random Early Detection (RED)

The random early detection algorithm works by randomly dropping packets with respect to the average queue length [6, 7]. When a packet arrives, the average queue length Q_{avg} is calculated. The calculation of the average queue length can vary depending of the implementation of the algorithm. A common practice is to calculate it based on the size of the previous average and the current size of the queue [8].

The average queue length is then compared with a certain configurable queue threshold (See Figure 5). If it is lower than the minimum threshold Q_{min}, the packet will be added to the queue. If it is between Q_{min} and the maximum threshold Q_{max}, the packet will either be dropped or queued depending on a certain probability. Following the increase of the average queue length,

doi: 10.2313/NET-2011-07-2_04

the drop probability grows linearly from zero to a specified maximum probability P_{max} at the point where $Q_{avg} = Q_{max}$. If the average queue length exceeds the Q_{max}, the packet will be dropped.

Note that RED will drop packet indiscriminately, meaning it has no mechanism to differentiate between traffic categories. In the case where the queue does become full, tail dropping is used.

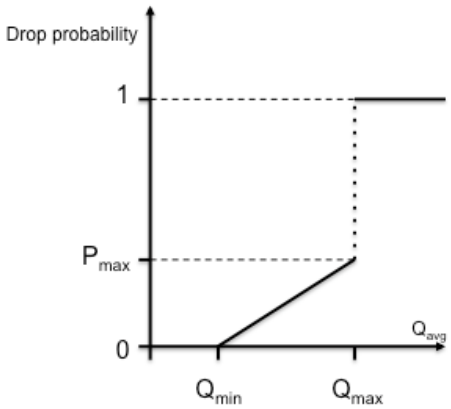

Figure 5: Random Early Detection

3.2 Weighted Random Early Detection (WRED)

The weighted RED is a variation of RED algorithm which supports multiple drop probability functions for each traffic category [6]. Principally traffic with higher drop precedence (e.g. P2P file-sharing traffic) will be discriminated against traffic with lower drop precedence (e.g. Real-time application traffic) or in other words packets with a lower priority (lower IP precedence) can be dropped more often than higher priority packets [5]. Note that all packets still share one single queue regardless of their precedence level.

Table 1: WRED Configuration

Precedence	Q_{min}	Q_{max}	MPD
0	12	20	5
1	14	20	5
2	16	25	5
3	18	25	5

Table 1 describes an example of a WRED configuration with four traffic precedence (traffic category, higher number means higher priority). The *mark probability denominator* (MPD) is here set to 5. This means that when the queue length is between Q_{min} and Q_{max} one out of five packets will be dropped (20% drop probability). With this configuration, packets with precedence level 0 will be randomly dropped once the queue size reaches 12 (12 packets are queued). In a similar manner, the algorithm will start randomly dropping packets with precedence level 2 once the queue size reaches 16. Once the queue size reaches 20, any incoming packets with precedence level 0 or 1 are dropped. For precedence level 2 or 3, packets are dropped once the queue size reaches 25.

The WRED configuration can be set up to be fully overlapped, partially overlapped or staggered [6] (See Figure 6).

The colors represent different traffic precedence (not to be confused with the three color marking scheme discussed in previously). It is worth noticing that only one single queue is being used regardless of the number of precedence levels. There is no way to control the composition of the queue, i.e. how many low priority or high priority packets there are in the queue. A fully overlapped configuration still provides a somewhat fair allocation of resource. Here we see that after a certain threshold all packet types could be dropped. Meanwhile a staggered configuration gives the advantage for high precedence packets at the cost of low precedence packets. Here we see that the minimum threshold for higher priority traffic is set higher than the maximum threshold of lower priority traffic. And due to the nature of the queue, it is possible that the queue is filled with only high priority packets up until that maximum threshold. From this point on all new lower priority packets will not have a chance to ever enter the queue.

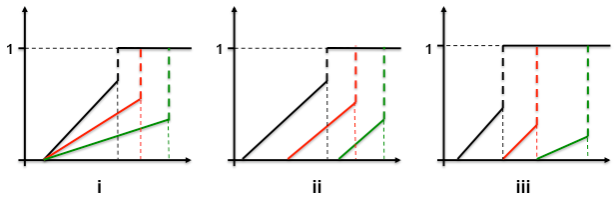

Figure 6: WRED configuration: i) fully overlapped, ii) partially overlapped, iii) staggered

3.3 RED with In/Out (RIO)

RED with In/Out queues is another extension of the RED algorithm, which uses separate (virtual) queue for each traffic precedence [1]. Incoming packets that comply with the contracted service profile are marked *In* and are added in the *In* queue. Those that do not comply with the service profile are marked *Out* and are added to the *Out* queue. It is so arranged, that during congestion the *Out* packets are dropped first. RIO can also be seen as WRED with two drop precedence, but maintain two separate queues, one for each drop precedence. This feature makes it possible to control the contribution of each precedence level in the total queue. In this respect, RIO is more appropriate to protect traffic with lower drop precedence against that with higher drop precedence [6].

3.4 Explicit Congestion Notification

The three congestion avoidance algorithms discussed previously work by dropping packets. There are two arguments that speak against this idea:

- Dropping packets while there is still available resource seems counter-intuitive, because "Why drop completely perfect packets when there is still free buffer space?"
- By dropping packet the network resource that was being used to deliver the packet up to the dropping node would be wasted.

doi: 10.2313/NET-2011-07-2_04

The RFC3168 [9] defines an extension to the Internet Protocol (IP) and to the Transmission Control Protocol (TCP) that allows an end-to-end congestion notification without the need to drop packets. This is called the *Explicit Congestion Notification* (ECN). For ECN to be effective, all nodes in along the route need to support this.

An end-point that supports ECN use the two rightmost bits of the DiffServ field in the IP header to mark that it is ECN capable (01 or 10, refer Table 2 for a complete list of the ECN bits). Although the ECN marking appears in the internet layer, ECN needs a transport protocol layer (or higher layer protocol) which not only supports the congestion control, but also has a way to deliver feedback about the occurring congestion to the transmitting end. TCP works fine because it has a congestion control mechanism and can support delivery of the feedback by using the ECE flag in the TCP header (refer to [9] for further details).

Table 2: ECN Bits

ECN Bit	***Meaning***
00	*Non-ECT* (Non ECN Capable Transport)
01	*ECT(1)* (ECN Capable Transport)
10	*ECT(2)* (ECN Capable Transport)
11	*CE* (Congestion Encountered)

When congestion occurs, an ECN capable router will set the ECN bit of an incoming ECN packet with 11 (Congestion Encountered) and still forward the packet (instead of dropping it). When the packet finally arrives, the transport layer of the receiving end will notice this congestion and will send feedback to the transmitting end. The latter will later be informed that congestion occurs and will adjust its transmission speed to avoid further congestion.

4. DIFFERENTIATED SERVICE

Differentiated Service (DiffServ) is the current accepted standard of implementing Quality of Service. It describes a computer network architecture that enable a simple and scalable mechanism to classify and manage network traffic. The RFC 2474 defines DiffServ as an enhancement to the internet protocol to enable scable service discrimination in the internet without the need for per-flow state and signaling at every hop. Services can be constructed by means of setting bits in an IP header (marking) at network boundaries, using those bits to determine how packets will be forwarded inside the core network, and conditioning the marked packets in accordance with the SLA [10].

DiffServ information is coded in the DiffServ field of the IP header (this is called the DiffServ codepoint). Differentiated services are realized by mapping the DiffServ codepoint to a particular forwarding behavior – the so called per-hop behaviors (PHB) at each node. The mapping also represent the classification of traffic. Traffic can be classified based on different parameters such as source address or destination address. Once a packet enters a DiffServ domain, it is subject to classification and conditioning. A packet entering a DiffServ domain may already have a DiffServ marking given by the DiffServ domain which forwarded it. A DiffServ domain may honor the previous marking, ignore it, or overwrite it.

Theoretically the six bits available for DiffServ codepoint allow network operators to specify up to 64 (2^6) different traffic classes (or PHB). However only four PHBs are commonly used. These are:

1. The default PHB. This PHB is defined in RFC 2474 and should be supported by all DiffServ capable domains. The default PHB is typically used for best-effort traffic.
2. Expedited Forwarding (EF) PHB. This PHB is defined in RFC 3246 [11]. It is usually used for low-loss and low-latency traffic.
3. Assured Forwarding (AF) PHB. This PHB is described in RFC 2597 [12]. It is used to guarantee packet delivery under certain conditions according to the AF class. Currently there are four AF classes specified, each with its own drop precedence.
4. Class selector PHB. Defined in RFC 2474, this PHB allows backward compatibility with the ToS bits of the earlier IP header specification for systems without DiffServ support (both ToS and DiffServ bits occupy the same location in the IP header).

5. CONCLUSION

Traffic shaping and congestion avoidance are techniques that can be used to increase performance and guarantee Quality of Services. They can be used in parallel. A typical scenario would be to monitor and mark packets at network edges, and set up an active queue management mechanism in the core network [13]. To protect traffic against future delay or congestion in core network, traffic shaping can be used.

When dealing with traffic shaping, there are some issues to be considered such as picking the correct values. Choosing CIR is a business decision. Choosing PIR in the other hand is a business decision with a technical impact [3]. As a link approaches its maximum capacity, the average packet latency becomes higher. When choosing bucket size, it is important to set it in respect to the maximum packet size that can be received on the link (otherwise it can never be forwarded, as it never gets the required amount of token, i.e. the policy is never met).

There are also issues to be considered when using active queue management. Generally all active queue management only performs well with responsive flows (such as TCP). With the co-existence of other non-responsive flows (such as UDP), dropping packet does not necessarily prevent hosts from sending too fast and thus congestion still cannot be avoided. This issue is discussed in [6]. Another issue is fairness across concurrent connection. A solution for this problem is proposed in [1].

Finally, maintaining a consistent Quality of Service is not an easy task to do. First it must be supported by all nodes along the path. Secondly we are also aware that packets may travel between multiple autonomous systems, which may have different QoS policy. In other words, a high priority traffic for one autonomous system might be just a low priority traffic for another one. To better ensure quality of service, a common understanding of traffic policing is needed. But again, this is not trivial.

doi: 10.2313/NET-2011-07-2_04

6. REFERENCES

[1] I. Andrikopoulos, L. Wood, and G. Pavlou. “A Fair Traffic Conditioner for the Assured Service in a Differentiated Service Internet”, Proceedings of IEEE International Conference on Communications ICC2000. 2000

[2] S. Blake, D. Black, M. Carlson, et. al. An Architecture for Differentiated Services. RFC 2475. 1998

[3] BTI Systems White Paper. Understanding Traffic Policing. 2010. http://www.btisystems.com/_documents/white-papers/Understanding-Traffic-Policing-WP0102.pdf

[4] J. Heinanen and R. Guerin. A Two Rate Three Color Marker. RFC 2698. 1999

[5] A. Balchunas. QoS and Congestion Avoidance. 2010. www.routeralley.com/ra/docs/qos_congestion_avoidance.pdf

[6] E. Bowen and C. Jeffries, L. Kencl, et. al. Bandwidth Allocation for a Non-Responsive Flows with Active Queue Management. 2002

[7] S. Floyd and V. Jacobson. “Random Early Detection Gateways for Congestion Avoidance”, ACM Transactions on Networking. August 1993

[8] Cisco Document. Distributed Weighted Random Early Detection. http://www.cisco.com/en/US/docs/ios/11_1/feature/guide/WRED.pdf

[9] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168. 2001

[10] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474.1998

[11] B. Davie and A. Charny, K. Benson, et. al. An Expedited Forwarding PHB. RFC 3246. 2002

[12] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB Group. RFC 2597. 1999

[13] Virpi Laatu, Jarmo Jarju and Pekka Loula. “The Impacts of Aggregation on the Performance of TCP Flows in DS Networks”, Proceedings of ICN2004. March 2004

doi: 10.2313/NET-2011-07-2_04