Computing and Informatics, Vol. 22, 2003, 317–335

A COMPARATIVE STUDY OF QUEUE, DELAY, AND LOSS CHARACTERISTICS OF AQM SCHEMES IN QOS-ENABLED NETWORKS

Jahwan Koo, Seongjin Ahn, Jinwook Chung

School of Information and Communications Engineering
Sungkyunkwan University
Chunchun-dong 300
Jangan-gu, Suwon, Kyounggi-do, Korea
e-mail: {jhkoo, jwchung}@songgang.skku.ac.kr, sjahn@comedu.skku.ac.kr

Manuscript received 2 September 2004; revised 15 November 2004
Communicated by Jiří Šafařík

Abstract. One of the major component in a QoS-enabled network is active queue management (AQM). Over the last decade numerous AQM schemes have been proposed in the literature. However, much recent work has focused on improving AQM performance through alternate approaches. This study focuses on an unbiased comparative evaluation of the various proposals. The evaluation methodology adopted is the following: we first define the relationship between the terminologies used in this paper, briefly introduce the queue, delay, and loss characteristics – a subset of network characteristics that can be used to describe the behavior of network entities, and give their mathematical description. Next, we present a method that would be a successful case study based on the *NS* simulation technique and simulation-based comparisons of AQM schemes chosen, which will help understand how they differ from in terms of per-node queueing information and per-flow end-to-end behavior. Simulation results showed that PI schemes, a feedback-based mechanism, can assist delay sensitive applications to adapt dynamically to underlying network and to stabilize the end-to-end QoS within an acceptable requirement. To understand this attribute and behavior is important for the proper design of queue disciplines, for the provisioning of queues and link capacity, and for choosing parameters in simulation.

Keywords: Network characteristic, simulation technique, queue schemes, per-node queueing, per-flow end-to-end behavior, Internet QoS

1 INTRODUCTION

Network technologies have been revolutionized in the last decade. The speed and capacity of various components in a networking system, such as transmission media, switches, and routers have been drastically increased. Millions of users and billions of traffic have introduced into wired and/or wireless networks. In addition, new peer-to-peer applications such as Napster, Kazza, and e-donkey as well as multimedia traffic such as audio and video have been widely delivered. For these reasons, network characteristics have been more complicated and diversified.

However, today's packet-switched networks only provide best-effort services. All traffic is processed as quickly as possible and treated in the same manner, but there is no guarantee as to performance requirements in terms of throughput, delay, delay jitter, and loss rate. The best-effort service is adequate as long as the applications using the network are not sensitive to variations in losses and delays, and if the load on the network is small. These conditions were true in the early days of the Internet, but do not hold anymore due to the increasing number of different applications using the Internet. There is an urgent need to provide network services with performance guarantees and to develop algorithms supporting different levels of services. The solutions are summarized as Quality-of-Service (QoS).

A QoS-enabled network is composed of various functions for providing different types of service to different packets such as rate controller, classifier, scheduler, and admission control. The scheduler function of them determines the order in which packets are processed at a node and/or transmitted over a link. The order in which the packets are to be processed is determined by the congestion avoidance and packet drop policy (also called *Active Queue Management*) at the node. AQM schemes have evolved over time and continue to do so.

Over the last decade numerous AQM schemes have been proposed in the literature. Although there are many papers related to AQM schemes, there have been few discussed together in a single paper. In this paper, we first define the relationship between the terminologies used in this paper, briefly introduce the queue, delay, and loss metrics of the main network characteristics that will be more complicated and diversified, and give their mathematical description. We also present a method that would be a successful case study based on the *NS* [25] simulation technique for describing network characteristics and simulation-based comparisons of AQM schemes, which will help understand how they differ from in terms of per-node queueing information and per-flow end-to-end behavior. To understand this attribute and behavior is important for the proper design of queue disciplines, for the provisioning of queues and link capacity, and for choosing parameters in simulation.

The rest of the paper is organized as follows. In Section 2, we define the terminologies used in this paper and network characteristics that can be used to describe the behavior of network components, investigate the metrics available to measure, present their mathematical descriptions, and provide related works. In Section 3, we present a description and model of the basic algorithms for QoS-enabled network including DropTail, Random Early Detection (RED) [14], BLUE [15], Random Ex-

ponential Marking (REM) [16], Proportional Integral (PI) [17], and Joint Buffer management and Scheduling (JoBS) [18]. In Section 4, we discuss why we use a network simulator and how we simulated network characteristics within the scope of this paper. Section 5 presents simulation-based comparisons of AQM schemes such as DropTail, RED, and PI. Its purpose is to identify the basic approaches that have been proposed and to classify them according to the design goals and performance issues of AQM schemes. The final section offers future work and concluding remark.

2 NETWORK CHARACTERISTICS

2.1 Definition of Terminologies

To describe network characteristics very well and to enable discussion and definition of network simulations, we will define the following terminologies, ranging from general to specific:

- A *network entity* is a general terminology that encompasses node, link (hop-by-hop) and path (end-to-end).
- *Network characteristics* are the intrinsic properties of a portion of the network that are related to the performance and reliability of the network.
- *Simulation methodologies* are the means and methods of evaluating those characteristics. Generally, there will be multiple ways to evaluate a given characteristic.
- A *monitored output* is an instance of the information obtained by applying the simulation methodology.

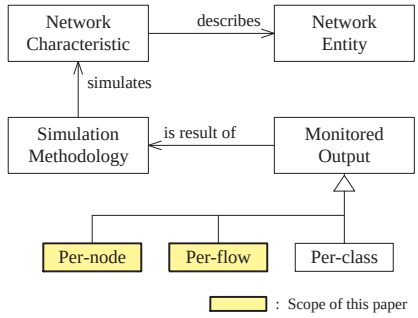


Fig. 1. The relationship between the terminologies used in this paper

The relationship between these terminologies is illustrated in Figure 1. We considered the terminology specified by the IETF Internet Protocol Performance Metrics(IPPM) Framework [1].

2.2 Definition of Network Characteristics

The characteristics hierarchy shown in Figure 2 is taken from Global Grid Forum (GGF) Network Measurements Working Group (NMWG) GFD-R-P.023 [2] that describes a standard set of network characteristics. The current hierarchy does not fully express all network characteristics, but they will be added to the hierarchy if needed. In this paper, we will focus only on queue, delay and loss characteristics, on discussing their definition and significance in terms of network characteristics, and on making a formal description of the metrics such as queue length, delay, average delay, and loss rate.

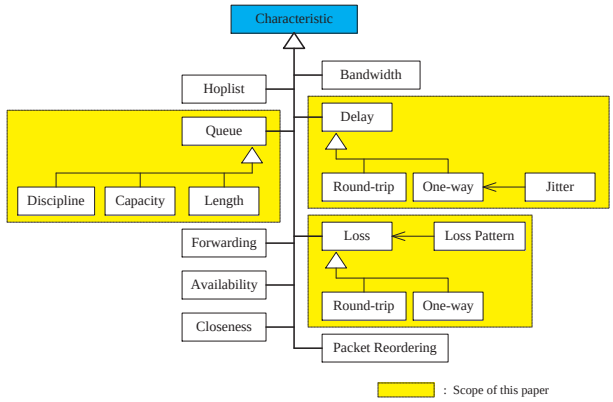


Fig. 2. A subset of network characteristics that can be used to describe the behavior of network entities

Queue: A queueing system in networks can be described as packets arriving for service, waiting for service if it is not immediate, and if having waited for service, leaving the system after being served. In most cases, four basic characteristics of queueing processes provide an adequate description of a queueing system in networks: (1) arrival pattern of packets, (2) service pattern of schedulers, (3) queue discipline, and (4) system capacity. To model network behavior, analytical models require information about the properties of the queues.

Delay: Delay is the time elapsed while a packet travels from one point (e.g., source premise or network ingress) to another (e.g., destination premise or network degress). As described in [3], delay is important because (1) some applications (e.g., audio and video applications) do not perform well (or at all) if end-to-end delay between nodes is large relative to some threshold value, (2) erratic variation in delay makes it difficult (or impossible) to support many real-time applications, (3) the larger the value of delay, the more difficult it is for transport-layer protocols to maintain high bandwidths, (4) the minimum value of this characteristic indicates the delay due only to propagation and transmission delay,

which will likely be experienced when the path traversed is lightly loaded, and (5) values of this characteristic above the minimum means the congestion present in the path. This characteristic can be specified in a number of different ways, including average delay, variance of delay (jitter), and delay bound.

Loss: Packets can be lost in a network because they may be dropped when a queue in the network node overflows. As described in [4], loss is important because (1) some applications (e.g., audio and video applications) do not perform well (or at all) if end-to-end loss between nodes is large relative to some threshold value, (2) excessive packet loss may make it difficult to support certain real-time applications, (3) the larger the value of packet loss, the more difficult it is for transport-layer protocols to maintain high bandwidths, (4) the sensitivity of real-time applications and of transport-layer protocols to loss become especially important when very large delay-bandwidth products must be supported, and (5) the sensitivity to loss of individual packets, as well as to frequency and patterns of loss among longer packet sequences is strongly dependent on the application itself. This characteristic can be specified in a number of different ways, including loss rate, loss patterns, loss free seconds, and conditional loss probability.

End-to-End Delay: Each packet generated by a source is traversed to the destination via a sequence of intermediate nodes. The end-to-end delay is thus the sum of the delays experienced at each node on the way to the destination. Each such delay in turn consists of two components, a constant component which includes the transmission delay at a node and the propagation delay on the link to the next node, and a variable component which includes the processing and queueing delays at the node.

End-to-End Loss Packets may be dropped at the intermediate nodes because of buffer overflow.

2.3 Mathematical Descriptions

Next, we give a formal description of the discussed metrics used in network characteristics. We assume that the link has a capacity C and a total queue space Q. We use $a(t)$, $l(t)$, and $r(t)$, respectively, to denote the arrivals, the amount of traffic dropped, and the service rate at time t.

We now introduce the notions of arrival curve, input curve, and output curve for a traffic in the time interval $[t_1, t_2]$. The arrival curve A and the input curve R^{in} are defined as

$$A(t_1, t_2) = \int_{t_1}^{t_2} \lambda(x)\mathrm{d}x, \tag{1}$$

where $\lambda(t)$ is the instantaneous arrival rate at time t, and

$$R^{in}(t_1, t_2) = A(t_1, t_2) - \int_{t_1}^{t_2} \xi(x)\mathrm{d}x, \tag{2}$$

where $\xi(t)$ is the instantaneous drop rate at time t. From Equation 2, the difference between the arrival and input curve is the amount of dropped traffic. The output curve R^{out} is the transmitted traffic in the interval $[t_1, t_2]$, given by

$$R^{out}(t_1, t_2) = \int_{t_1}^{t_2} r(x)\mathrm{d}x, \tag{3}$$

where $r(t)$ is the instantaneous service rate at time t. From now on, we will use the following shorthand notations to denote the arrival, input and output curves at a given time t, respectively:

$$A(t) = A(0, t),$$
$$R^{\text{in}}(t) = R^{in}(0, t),$$
$$R^{\text{out}}(t) = R^{out}(0, t).$$

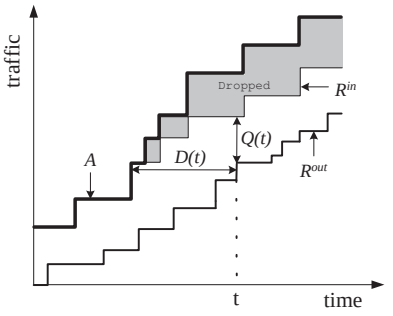


Fig. 3. Queue length, delay and loss

In Figure 3, the vertical and the horizontal distance between the input and output curves, respectively, are the queue length Q and the delay D. The delay D at time t is the delay of an arrival which is transmitted at time t. Queue length and delay at time t are defined as

$$Q(t) = R^{\text{in}}(t) - R^{\text{out}}(t), \tag{4}$$

and

$$D(t) = \max_{x<t}\{x | R^{\text{out}}(t) \geq R^{\text{in}}(t - x)\}. \tag{5}$$

We also obtain the average delay by averaging the instantaneous delay $D(t)$ over a time window of length τ.

$$D_t^{\text{avg}}(\tau) = \frac{1}{\tau}\int_{t-\tau}^{t} D(x)\mathrm{d}x. \tag{6}$$

We denote the loss rate as $P(t)$, which expresses the fraction of lost traffic since the beginning of the current busy period at time t_0. A busy period is a time interval

with a positive queue length of traffic. So, $P(t)$ expresses the fraction of traffic that has been dropped in the time interval $[t_0, t]$, that is,

$$P(t) = \frac{\int_{t0}^{t} l(x)\mathrm{d}x}{\int_{t0}^{t} a(x)\mathrm{d}x} = 1 - \frac{R_{in}(t_0, t^-) + a(t) - l(t)}{A(t_0, t)}. \tag{7}$$

with

$$t^- = \sup\{x | x < t\}.$$

With the metrics just defined, therefore, we showed that we can now formally describe network characteristics.

2.4 Related Work

To understand network characteristics is very important for designing, implementing, maintaining, and provisioning network components. Many studies of network characteristics in various network environments have been reported in the literature, e.g., [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18].

In general, the methodology to describe network characteristics is divided into (1) the analytic approaches [5, 6, 7, 8] which have been used in queueing network models and analyzed corresponding models simply with certain assumptions, (2) the experimental approaches [9, 10, 11] which have monitored the variations of network characteristics on a given environment and/or measured the metrics passively or actively, and (3) the simulation approaches [12, 13, 14, 15, 16, 17, 18] which have been used to design a model of a real system and conducted experiments with this model for the purpose either of understanding the behavior of the system or evaluate various strategies for the operation of the system. Also, it has been suggested [22] that an advance in simulation approaches is the birth of network simulator, *NS* [25], which is indeed a remarkable thing developed by Virtual Network Testbed (VINT) project. Especially, many researchers have used *NS* to develop and investigate AQM schemes discussed in this paper as well as to identify their behavior and characteristics under varied conditions [14, 15, 16, 17, 18].

AQM schemes can be classified in different ways. In [19], these were classified based on two dimensions: one is when decision on discarding packets is made and the other is what information is used to make packet discard decisions. In [20], a detailed classification of the various schemes has been proposed. It was based on the networking environment (ATM or IP networks), the type of congestion management mechanism implemented (congestion avoidance or congestion control and recovery), the number of thresholds used (none, global, or per-connection), decision information (global or per-connection), and the queue behavior (static or dynamic).

An important factor in IP network to provide the required QoS is to avoid the effects of network congestion. Congestion occurs on a communication link whenever the amount of traffic injected on that link exceeds its capacity. This excess traffic causes queuing delays to increase rapidly as buffers fill up, and in extreme cases can cause the buffers to overflow, losing packets. Thus it is crucial to have some kind

of congestion management, of which there are two broad approaches: congestion control and recovery, and congestion avoidance. While the first approach is reactive and enables the network to recover when congestion has occurred, the second approach is proactive and attempts to avoid congestion in the first place. Because of the existence of different traffic classes in a single network, typically both reactive and proactive methods are applied. Congestion avoidance methods apply open-loop and closed-loop flow control mechanisms and require the cooperation of the end systems. Reservation of resources, admission control, and traffic policing are among the most commonly used open-loop mechanisms to avoid congestion. Closed-loop mechanisms, on the other hand, rely on feedback to detect and prevent congestion where the feedback can be explicit, implicit, window-based, rate-based, end-to-end, and hop-by-hop. Congestion control and recovery mechanisms are applied within the network to combat congestion and they do not require the collaboration of the end systems. The most common way to combat congestion is by dropping packets whenever there is insufficient queue space.

Criteria	Congestion		Thresholds			State-Info	
Schemes	Control	Avoidance	None	Global	Per-Conn	Global	Per-Conn
DT	•		•			•	
RED		•		•		•	
BLUE		•		•		•	
REM		•		•		•	
PI		•			•		•
JoBS		•			•		•

Table 1. Classification of AQM Schemes

In this paper, a wide variety of AQM schemes with many different characteristics for IP network are considered, and hence a more extensive classification is proposed in Table 1. We refer to a survey article [20, 23, 24] for an exhaustive review of all possible AQM schemes.

Initial proposals for AQM for IP network [14, 15, 16] were motivated by the need to improve TCP performance, without considering service differentiation. More recent research efforts [17, 18] enhance these initial proposals in order to provide service differentiation.

3 AQM SCHEMES

We provide a description of the basic schemes for IP network including DropTail, RED [14], BLUE [15], REM [16], PI [17], and JoBS [18] and present analytic models of their dropping (or marking) probability.

3.1 DropTail

DropTail maintains exactly simple FIFO queues. There are no methods, configuration parameter, or state variables that are specific to drop tail queues.

3.2 RED

RED [14] was presented with the objective to minimize packet loss and queueing delay, to avoid global synchronization of sources, to maintain high link utilization, and to remove biases against bursty sources. To achieve these goals, RED utilizes two thresholds, $\min_{th}$ and $\max_{th}$, and a exponentially-weighted moving average (EWMA) formula to estimate the average queue length, $Q_{avg} = (1 - W_q) \cdot Q_{avg} + W_q \cdot Q$, where Q is the current queue length and W_q is a weight parameter, $0 \leq W_q \leq 1$. The two thresholds are used to establish three zones. If the average queue length is below the lower threshold ($\min_{th}$), RED is in the normal operation zone and all packets are accepted. On the other hand, if it is above the higher threshold ($\max_{th}$), RED is in the congestion control region and all incoming packets are dropped. If the average queue length is between both thresholds, RED is in the congestion avoidance region and the packets are discarded with a certain probability P_a:

$$P_a = \frac{P_b}{(1 - count \cdot P_b)}. \tag{8}$$

This probability is increased by two factors. A counter is incremented every time a packet arrives at the router and is queued, and reset whenever a packet is dropped. As the counter increases, the dropping probability also increases. In addition, the dropping probability also increases as the average queue length approaches the higher threshold. In implementing this, RED computes an intermediate probability P_b,

$$P_b = \frac{\max_p}{\max_{th} - \min_{th}} \cdot (Q_{avg} - \min_{th}) \tag{9}$$

whose maximal value given by $\max_p$ is reached when the average queue length is equal to $\max_{th}$. For a constant average queue length, all incoming packets have the same probability to get dropped. As a result, RED drops packets in proportion to the connections' share of the bandwidth.

3.3 BLUE

BLUE [15] uses different metrics to characterize the probability of dropping an arrival. This scheme uses the current loss ratio and link utilization as input parameters. It maintains a single probability, P_m, which it uses to mark (or drop) packets when they are enqueued. If the queue is continually dropping packets due to queue overflow, BLUE increments P_m,

$$P_m = P_m + d1, \tag{10}$$

thus increasing the rate at which it sends back congestion notification. Conversely, if the queue becomes empty or if the link is idle,

$$P_m = P_m - d2, \tag{11}$$

BLUE decreases its marking probability. This effectively allows BLUE to learn the correct rate it needs to send back congestion notification. Beside the marking probability, BLUE uses two other parameters which control how quickly the marking probability changes over time. The first is *freeze time*. This parameter determines the minimum time interval between two successive updates of P_m. This allows the changes in the marking probability to take effect before the value is updated again. The other parameters used (d_1 and d_2), determine the amount by which P_m is incremented when the queue overflows or is decremented when the link is idle.

3.4 REM

REM [16] is an AQM scheme that measures congestion not by a performance measure such as loss or delay, but by a quantity we call "price". It maintains "price" as a congestion measure to determine the marking probability, p_l, based on rate mismatch (i.e., difference between input rate and link capacity) and queue mismatch (i.e., difference between queue lenght and target value q_{ref}.

$$p_l(t+1) = \max(0, p_l(t) + \gamma(\alpha_l(q_l(t) - q_{ref}) + x_l(t) - c_l(t))) \tag{12}$$

The cost is directly proportional to the queue occupancy.

3.5 PI

PI [17] uses a feedback-based model for TCP arrival rates to let the queue occupancy converge to a target value, but assumes a priori knowledge of the round-trip times and of the number of flows traversing the router. It improves responsiveness of the TCP flow mechanisms by means of proportional control, stabilizes the queue length around target value q_{ref} by means of integral control, and marks each packet with a probability, p,

$$p(t+1) = p(t) + a(q(t+1) - q_{ref}) - b(q(t) - q_{ref}). \tag{13}$$

Two main functions are used in the PI algorithm: one is the congestion indicator (to detect congestion) and the other is the congestion control function (to avoid and control congestion). The PI-controller has been designed based on (1) not only to improve responsiveness of the TCP flow dynamics but also to stabilize the router queue length around Q_{ref}. The latter can be achieved by means of integral (I)-control, while the former can be achieved by means of proportional (P)-control using the instantaneous queue length rather than using the exponentially weighted moving average (EWMA) queue length.

3.6 JoBS

JoBS [18] is capable of supporting a wide range of relative, as well as absolute, per-class guarantees for loss and delay, without assuming admission control or traffic policing.

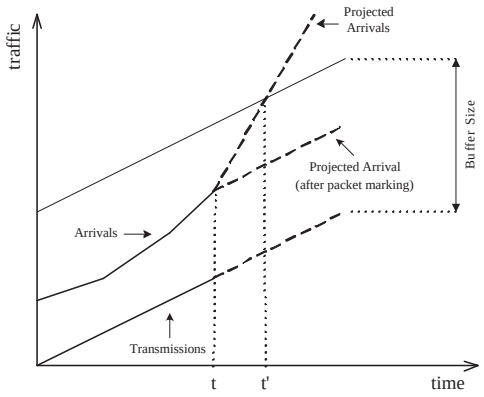


Fig. 4. The concepts of JoBS algorithm

The main idea of the JoBS scheme is as follows: At time t of a packet arrival, the router estimates the congestion window size, $W(t)$, and the round-trip time, $RTT(t)$, of the TCP flow. With these estimates, future traffic arrivals are projected, and impending queue overflows are inferred. If a packet loss is projected, JoBS reduces the congestion window size of the TCP source by marking packets with Explicit Congestion Notification (ECN). By reducing the congestion window size, the sending rate of the TCP source is reduced, and impending packet losses can be avoided. At any time t, the backlog at the router is equal to $R^{\text{in}}(t) - R^{\text{out}}(t)$. Hence, the JoBS makes an effort to meet the following requirement:

$$\forall t : R^{\text{in}}(t) - R^{\text{out}}(t) \leq B_{\text{lim}} \tag{14}$$

where B_{lim} is the size of the router's buffer, $R^{\text{in}}(t)$ is the total amount of traffic that has entered the router until time t, and $R^{\text{out}}(t)$ is the total amount of traffic that has left the router until time t.

4 SIMULATION METHODOLOGY

In this section, we discuss why we use a network simulator and how we simulated network characteristics discussed in the previous section.

4.1 Simulation Package Selection

We will choose a network simulation package because (1) we want verifiable and reproducible results, and (2) we want to test network characteristics in a variety of conditions.

The *NS-2* [25] we chose is an object-oriented, discrete event driven network simulator written in C++ and OTcl that simulates variety of IP networks. It has the following main functionalities that can be used for describing network characteristics. First, arbitrary network topologies, composed of nodes, links and paths can be defined. It also implements traffic models and applications such as ftp, telnet, web, constant-bit rate, variable-bit rate and real audio, transport protocols such as TCP (Reno, Vegas, etc.) and UDP, queue disciplines such as DropTail, RED, priority and fair queue, routing algorithms such as link state and distance vector, and more. Second, it is based on two object-oriented languages (one is OTcl script interpreter, another is C++ compiler). This leads to a better reusability, efficiency and maintenance. Third, there are two primary but distinct types of monitoring capabilities currently supported by the simulator. The first, called *traces*, record each individual packet as it arrives, departs, or is dropped at a link or queue. The other types, called *monitors*, record counts of various interesting quantities such as packet and byte arrivals, departures, etc. Monitors can also monitor counts associated with all packets, or on a per-flow basis.

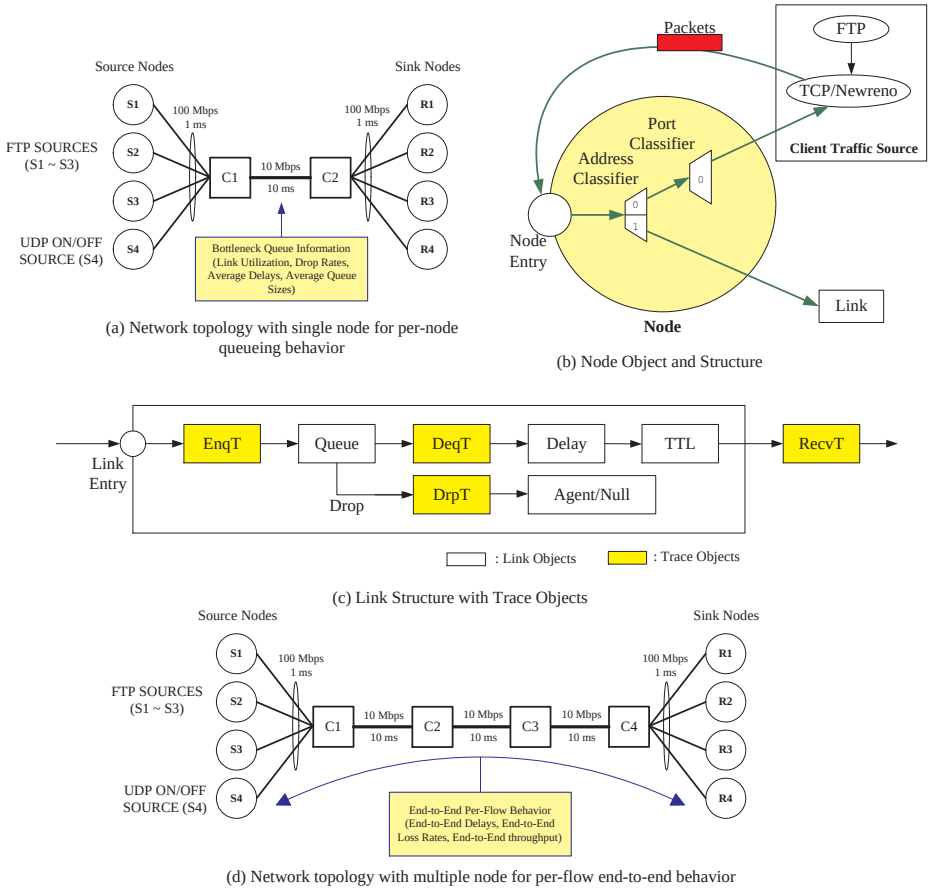
4.2 Simulation Goal and Network Topology

We perform two simulation experiments. In the first experiment, we compare the performance provided by the DropTail, RED, PI schemes, at a single node. The first experiment focuses on the performance issues of the queueing information at a bottleneck link. In the second experiment, we augment the first experiment's network topology to the multiple nodes with TCP and UDP traffic, and assess the per-flow end-to-end characteristics.

4.2.1 Setting Up Single-node Topology for Experiment 1

We consider a bottleneck link with 10 Mbps bandwidth, 10 ms propagation delay, and 15 0000 bytes queue size as shown in Figure 5 a). The rest of the links (edge links) are all 100 Mbps. Their propagation delay is 1 ms. Each source node is connected to the corresponding sink node at the other side of the network, i.e., source node Si is connected to sink node Ri. A node is a compound object composed of a node entry object and classifiers as shown in Figure 5 b). Similarly, a link is another major compound object as shown in Figure 5 c). Since there are numerous tutorials as well as manuals related the node and link object in *NS-2*, we will not pursue further discussion of this subject. There are 3 TCP source/sinks and one UDP source/sink connected to each edge node. Each TCP source is an FTP application on top of NewReno TCP. The FTP packet size is 500 bytes. Each UDP source is a Pareto

On-Off source with peak rate 5 000 Kbps, burst time 10 ms, and idle time 10 ms. The shape parameter α of the Pareto distribution essentially characterizes the burstiness of the traffic arrivals. The smaller α, the burstier the traffic. The experiment lasts for 70 seconds of simulated time, and ECN is available in the entire network.



(a) Network topology with single node for per-node queueing behavior

(b) Node Object and Structure

(c) Link Structure with Trace Objects

(d) Network topology with multiple node for per-flow end-to-end behavior

Fig. 5. Network topology, node, and link

DropTail. We use DropTail to have an estimate of the performance measure encountered without AQM scheme. With DropTail queue, incoming packets are discarded only when the queue is full.

RED. RED utilizes two thresholds, $\min_{th}$ and $\max_{th}$, and an EWMA formula to estimate the average queue length. RED is configured with a minimum threshold $\min_{th} = 30\,000$ bytes, and a maximum threshold $\max_{th} = 120\,000$ bytes. Also,

the parameter $\max_p$ is set to 1, and the weight used in the computation of the average queue size is set to $W_q = 0.002$.

PI. We configure the PI algorithm with approximate RTTs and a tight upper bound on the round-trip times $R+ = 180$ ms, with a sampling frequency of 160 Hz, and get $a = 1.643e{-}4$ and $b = 1.628e{-}4$. The target queue length Q_{ref} is set to 70 000 bytes. Note that such a crude parameter tuning is to account for the uncertainty on estimates of the RTTs and of the number of flows at router configuration time.

4.2.2 Setting Up Multiple Node Topology for Experiment 2

We use the same traffic pattern as in experiment 1 and augment the single-bottleneck core node to multiple core nodes as shown in Figure 5 d).

4.3 OTcl Script for Traffic Sources and Tracing

To setup and run a simulation network, we should write an OTcl script that initiates an event scheduler, sets up the network topology using the network objects and the functions in the library, and tells traffic sources when to start and stop transmitting packets through the event scheduler. We present an OTcl script for generating traffic sources (TCP/FTP and UDP/Pareto) and creating trace file as shown in Figure 6. Note that we have used *int-ftp* and *build-pareto-on-off* procedures to describe long file transfer and busty UDP traffic, respectively, which can be representatives of traffic pattern in the Internet.

5 SIMULATION RESULTS

In this section, we present simulation results for each queue discipline in terms of per-node queueing behavior and per-flow end-to-end behavior. In referencing [21], however, this paper will be limited to simulation results. In summary, queue disciplines have mainly focused on seven issues: 1) avoid congestion, 2) reduce the packet transfer delay, keeping the queue lengths at low levels, 3) avoid the TCP global synchronization problem, 4) achieve fairness among different traffic types, 5) deliver service guarantee (guaranteed or differentiated), 6) reduce the program complexity, and 7) increase the scalability. These issues, however, are all inter-related.

5.1 Per-Node Queueing Behavior

For each discipline, we monitor link utilization, loss rate, average delay, and average queue length at the bottleneck core link as shown Figure 5 a), and present our results in Figure 7. Note that RED is the worst of the three schemes with regard to link utilization and packet drop rates. This result makes us understand that RED discipline has a heavy effect on the parameter configuration.

```
##### Traffic Sources Primitives #####
proc flow_setup {id src src_agent dst dst_agent
app_flow pksize}
{
    global ns
    $ns attach-agent $src $src_agent
    $ns attach-agent $dst $dst_agent
    $ns connect $src_agent $dst_agent
    $src_agent set packetSize_ $pksize
    $src_agent set fid_ $id
    $app_flow set flowid_ $id
    $app_flow attach-agent $src_agent
    return $src_agent
}

proc inf_ftp {id src src_agent dst dst_agent app_flow
maxwin pksize starttm}
{
    global ns
    flow_setup $id $src $src_agent $dst $dst_agent \\
    $app_flow $pksize
    $src_agent set window_ $maxwin
    $src_agent set ecn_     1
    $ns at $starttm "$app_flow start"
    return $src_agent
}

proc build-pareto-on-off {id src src_agent dst dst_agent
app_flow peak_rate pksize shape_par start_tm idle_tm burst_tm}
{
    global ns
    flow_setup $id $src $src_agent $dst $dst_agent $app_flow \\
    $pksize
    $app_flow set burst-time_   [expr $burst_tm  / 1000.]
    $app_flow set idle-time_    [expr $idle_tm  / 1000.]
    $app_flow set rate_         [expr $peak_rate * 1000.0]
    $app_flow set shape_        $shape_par
    $ns at $start_tm "$app_flow start"
}
```

```
##### Global Variables #####

# Packet Size(in bytes) and TCP Window Size
set PKTSZ       500
set MAXWIN      50
# Number of monitored flows
set N_CI4
set N_USERS     4
set N_USERS_TCP3
# Alpha parameter of the Pareto distribution
set PARETO_ALPHA_C      1.3
set START_TM    0.0

##### Main #####

set ns [new Simulator]
set rnd [new RNG]
set seed 16
$rnd seed $seed
set nf [open dt/out.nam w]

# Trace File
$ns namtrace-all $nf

# Generating User Traffic (TCP/FTP and UDP Pareto)
for {set i 1} {$i <= $N_USERS} {incr i}
{
    if {$i <= $N_USERS_TCP} {
        set flow($i) [new Application/FTP]
        inf_ftp $i $source($i) $source_agent($i) $sink($i) \\
        $sink_agent($i)
        $flow($i) $MAXWIN $PKTSZ $START_TM
    } else {
        set flow($i) [new Application/Traffic/Pareto]
        build-pareto-on-off $i $source($i) $source_agent($i) \\
        $sink($i)
        $sink_agent($i) $flow($i) 5000.0 $PKTSZ $PARETO_ALPHA_C
        $START_TM 10.0 10.0
    }
}
```

Fig. 6. OTcl script for generating traffic sources and creating trace file

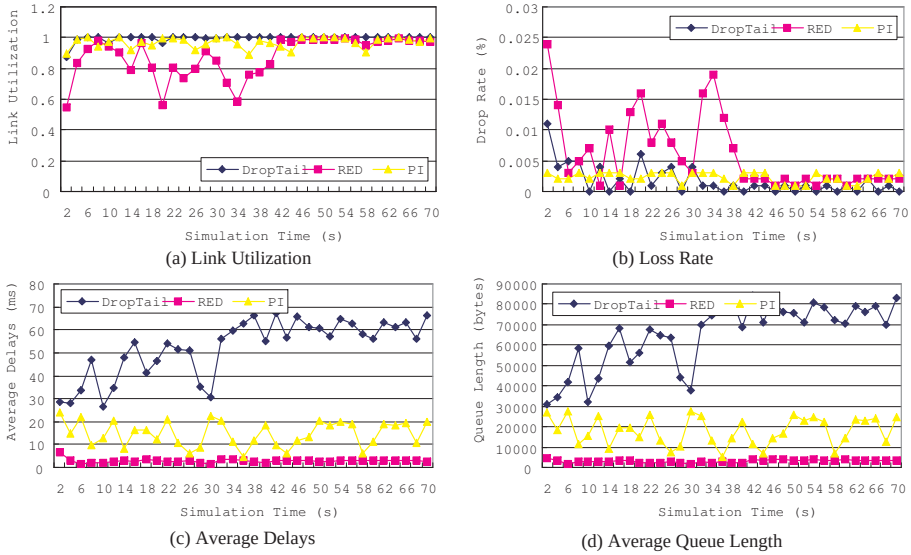


Fig. 7. Per-node queueing behavior with single node

5.2 Per-Flow End-to-End Behavior

For each discipline, we monitor end-to-end packet delays, end-to-end packet loss rates, and end-to-end throughput as shown Figure 5 d), and present only an end-to-end delay result in RED as shown Figure 8. Each of these four flows traverses four nodes, each node providing a queueing delay. Adding to this per-node queueing delay the propagation delays between each node, one can infer that the end-to-end delays of a flow in RED scheme have to be less than $4 \times D_t^{avg} + 3 \times 10 + 2 \times 1$.

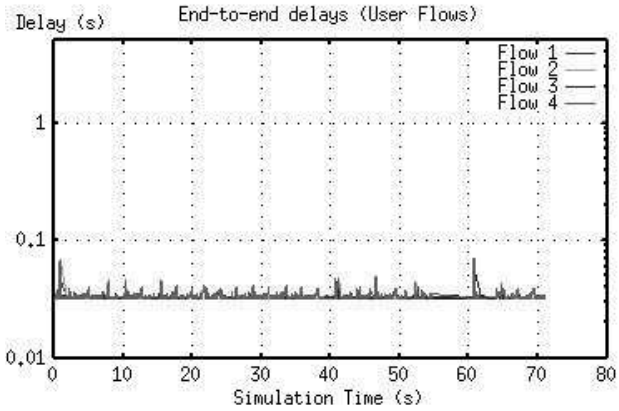


Fig. 8. Per-flow end-to-end delay with multiple nodes in RED

Note that PI scheme can achieve lower-delay performance than the remaining schemes. Furthermore, PI scheme was almost unaffected shown in the standard deviation of Table 2. It means that PI, a suitable feedback mechanism, can assist delay sensitive applications to adapt dynamically to underlying network and to stabilize the end-to-end QoS within an acceptable requirement. We have shown that the equilibrium and dynamics of the network depends on AQM schemes. In our opinion, a proper congestion signalling from network will be an effective solution to the instantaneous network fluctuation in the Internet.

AQM Schemes	Average		Standard Deviation	
	Loss Rate (%)	Delay (ms)	Loss Rate (%)	Delay (ms)
DT	0.174	52.27	0.238	12.458
RED	0.171	51.07	0.230	11.454
PI	0.226	14.67	0.082	5.487

Table 2. End-to-end performance results of AQM schemes

6 FUTURE WORK AND CONCLUSION

In this paper, we defined and classified various network characteristics, and showed that we can describe them through the simulation approach. Simulation results showed that PI schemes, a feedback-based mechanism, can assist delay sensitive applications to adapt dynamically to underlying network and to stabilize the end-to-end QoS within an acceptable requirement. The analysis has two objectives. First, we describe each algorithm's design goals and performance issues. Second, we compare the performance of our surveyed AQM schemes in QoS-enabled networks. In addition, the method presented in this paper could be used as a common simulation methodology to identify the behavior of network entities that will be more complicated and diversified. Therefore, we need to be supplemented with the ability of simulations to effectively describe, analyze, and evaluate network characteristics.

We are currently describing another network characteristics discussed in this paper by using the similar method. Also, we will extend the network topology toward a Grid-enabled network. Despite all the efforts and results described above, the analysis of network characteristics remains an area which deserves more research attention as far as network dynamics exist.

REFERENCES

[1] Paxson, V.—Almes, G.—Mahdavi, J.—Mathis, M.: Framework for IP Performance Metrics. RFC2330, May 1998.

[2] Lowekamp, B.—Tierney, B.—Cottrell, L.—Hughes-Jones, R.—Kielmann, T.—Swany, M.: A Hierarchy of Network Performance Characteristics for Grid Applications and Services. GFD-R-P.023, May 2004.

[3] Almes, G.—Kalidindi, S.—Zekauskas, M.: A One-Way Delay Metric for IPPM. RFC2679, September 1999.

[4] Almes, G.—Kalidindi, S.—Zekauskas, M.: A One-Way Packet Loss Metric for IPPM. RFC2680, September 1999.

[5] Cidon, I.—Khamisy, A.—Sidi, M.: Analysis of Packet Loss Processes in High-Speed Networks. IEEE Trans. Info. Theory, Vol. 39, 1993, No. 1, pp. 98–108.

[6] Low, S. H.—Srikant, R.: A Mathematical Framework for Designing a Low-Loss, Low-Delay Internet, Network and Spatial Economics. 2004.

[7] Cruz, R. L.: A Calculus for Network Delay, Part I: Network Elements in Isolation. IEEE Trans. Info. Theory, Vol. 37, 1991, No. 1, pp. 114–131.

[8] Cruz, R. L.: A Calculus for Network Delay, Part II: Network Analysis. IEEE Trans. Info. Theory, Vol. 37, 1991, No. 1, pp. 132–141.

[9] Mills, D.: Internet Delay Experiments. RFC 889, December 1983.

[10] Claffy, K.—Polyzos, G.—Braun, H.-W.: Traffic Characteristics of the T1 NSFNET Backbone. Proc. IEEE Infocom '93, San Fransisco, CA, pp. 885–892, April 1993.

[11] Sanghi, D.—Agrawala, A. K.—Jain, B.: Experimental Assessment of End-to-End Behavior on Internet. Proc. IEEE Infocom '93, San Fransisco, CA, pp. 867–874, March 1993.

[12] Zhang, L.—Clark, D. D.: Oscillating Behavior of Network Traffic: A Case Study Simulation, Internetworking: Research and Experience, Vol. 1, 1990, No. 2, pp. 101–112.

[13] Zhang, L.—Shenker, S.—Clark, D. D.: Observations on the Dynamics of a Congestion Control Algorithm: The Effects of 2-Way Traffic. Proc. ACM Sigcomm '91, Zurich, Switzerland, pp. 133–147, September 1991.

[14] Floyd, S.—Jacobson, V.: Random Early Detection for Congestion Avoidance. IEEE/ACM Transactions on Networking, Vol. 1, 1993, No. 4, pp. 397–413.

[15] Feng, W.—Kandlur, D.—Saha, D.—Shin, K.: Blue: A New Class of Active Queue Management Algorithms. Technical Report CSE-TR-387-99, University of Michigan, April 1999.

[16] Athuraliya, S.—Li, V. H.—Low, S. H.—Yin, Q.: REM: Active Queue Management. IEEE Network, Vol. 15, Issue 3, pp. 48–53, May 2001.

[17] Hollot, C. V.—Misra, V.—Towsley, D.—Gong, W.: On Designing Improved Controllers for AQM Routers Supporting TCP Flows. In Proceedings of IEEE INFOCOM 2001, Vol. 3, 2001, pp. 1726–1734, Anchorage, AK.

[18] Liebeherr, J.—Christin, N.: Buffer Management and Scheduling for Enhanced Differentiated Services. Technical Report CS-2000-24, University of Virginia, 2000.

[19] Guerin, R.—Peris, V.: Quality-of-Service in Packet Networks: Basic Mechanisms and Directions. Computer Networks, Vol. 31, 1999, pp. 169–189.

[20] Labrador, M.—Banerjee, S.: Packet Dropping Policies for ATM and IP Networks. IEEE Communications Surveys, Vol. 2, 1999, No. 3, pp. 2–14.

[21] Koo, J.—Ahn, S.—Chung, J.: Performance Analysis of Active Queue Management Schemes for IP Network. 4th International Conference, Krakow, Poland, LNCS 3036, pp. 349–356, June 6–9, 2004.

[22] Breslau, L.—Estrin, D.—Fall, K.—Floyd, S.—Heidemann, J.—Helmy, A.—Huang, P.—McCanne, S.—Varadhan, K.—Xu, Y.—Yu, H.: Advances in Network Simulation. IEEE Computer, Vol. 33, 2000, No. 5, pp. 59–67.

[23] Bitorika, A.—Robin, M.—Huggard, M.: A Survey of Active Queue Management Schemes. Trinity College Dublin, Department of Computer Science, Tech. Rep., Sept. 2003.

[24] Bitorika, A.—Robin, M.—Huggard, M.: An Evaluation Framework for Active Queue Management Schemes. in Proc. MASCOTS '03.

[25] NS-2 Network Simulator. `http://www.isi.edu/nsnam/ns/`.

Jahwan Koo received the B.S. and M.S. degrees in information engineering from Sung-kyunkwan University, Suwon, Korea, in 1995 and 1997, respectively. He is currently working towards the Ph.D. degree in information communication engineering with the School of Information and Communications Engineering, Sungkyunkwan University, Suwon, Korea. For more than five years, he was an infrastructure architect at LG CNS Co., Ltd., Seoul, Korea. His research interests include quality of services in IP network, network management, network security, and information technology architecture.

Seongjin Ahn received the B.S., M.S., and Ph.D. degrees in information and communication engineering from Sungkyunkwan University, Suwon, Korea, in 1988, 1990, and 1998, respectively. For more than five years, he was a Researcher in Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. He is currently an assistant professor of computer education, Sungkyunkwan University, Seoul, Korea. His research interests include network management, network security, and information assurance.

Jinwook Chung received the B.S. degrees in electric engineering from Sungkyunkwan University, Seoul, Korea, in 1974, the M.S. degrees in electronic engineering from Sung-kyunkwan University, Seoul, Korea, in 1977, and the Ph.D. degree in computer science from Seoul National University, Seoul, Korea, in 1991. For more than ten years, he was a section chief in Electronics and Telecommunications Research Institute (ETRI), Daejeon, Korea. Since 1984 he has been a professor of the School of Information and Communications Engineering, Sungkyunkwan University, Suwon, Korea. In 2002 he served as President of the Korea Information Processing Society (KIPS). His research interests include data communication, computer networks, network management, and network security. He has guided more than 150 M.S./Ph.D. students in this area of study and has published more than 100 papers in technical journals and conference proceedings.