International Journal of Computer Networks (IJCN)

ISSN : 1985 - 4129

Volume 1 - Issue 1
Number of issues per year: 6

Editor in Chief Associate Professor Min Song

International Journal of Computer Network (IJCN)

Book: 2009 Volume 1, Issue 1
Publishing Date: 31-11-2009
Proceedings
ISSN (Online): 1985-4129

IJCN Journal is a part of CSC Publishers
http://www.cscjournals.org

Typesetting: Camera-ready by author, data conversation by CSC Publishing Services – CSC Journals, Malaysia

CSC Publishers

Table of Contents

Volume 1, Issue 1, November 2009.

Deterministic Formulization of End-to-End Delay and Bandwidth Efficiency for Multicast Systems

Syed S. Rizvi srizvi@bridgeport.edu
Computer Science and Engineering Department
University of Bridgeport
Bridgeport, CT 06601, USA

Aasia Riasat aasia.riasat@iobm.edu.pk
Computer Science Department
Institute of Business Management
Karachi, 78100, Pakistan

Khaled M. Elleithy elleithy@bridgeport.edu
Computer Science and Engineering Department
University of Bridgeport
Bridgeport, CT 06601, USA

Abstract

End-System multicasting (ESM) is a promising application-layer scheme that has been recently proposed for implementing multicast routing in the application layer as a practical alternative to the IP multicasting. Moreover, ESM is an efficient application layer solution where all the multicast functionality is shifted to the end users. However, the limitation in bandwidth and the fact that the message needs to be forwarded from host-to-host using unicast connection, and consequently incrementing the end-to-end delay of the transmission process, contribute to the price to pay for this new approach. Therefore, supporting high-speed real-time applications such as live streaming multimedia, videoconferencing, distributed simulations, and multiparty games require a sound understanding of these multicasting schemes such as IP multicast and ESM and the factors that might affect the end-user requirements. In this paper, we present both the analytical and the mathematical models for formalizing the end-to-end delay and the bandwidth efficiency of both IP and ESM multicast system. For the sake of the experimental verifications of the proposed models, numerical and simulation results are presented in this paper. Finally, the proposed formulization can be used to design and implement a more robust and efficient multicast systems for the future networks.

Keywords: Bandwidth Analysis, End-to-End Delay, End System Multicast, IP Multicast, Overlay Networks.

1. INTRODUCTION

There is an emerging class of Internet and Intranet multicast applications that are designed to facilitate the simultaneous delivery of information from a single or multiple senders to multiple receivers. Different approaches of multicasting have been suggested to improve the overall performance of networks especially the Internet [1, 2, 3, 4]. These approaches are: multiple unicast, IP multicast, and end-system multicast. All of these methods have some advantages and disadvantages but the last two approaches (IP multicast, and end-system multicast) mentioned above have had more research effort in terms of performance evaluation of networks. Multiple unicast can be described as a service where one source sends the same copy of the message to multiple destinations. There is a one to one connection all the way from the source to the destination. No special configuration is required. In IP multicast, one source sends data to a specific group of receivers. In this case, a unique and special IP address is used, a class D address for the entire group. In addition, there is a special configuration adopted for efficiency reasons. A tree rooted at the source is constructed and only one copy of the message is sent since the routers along the paths to the destinations performed the necessary replication functionalities.

Finally, the end-system multicasting is a very promising application layer solution where all the multicast functionality is shifted to the end users [7]. In an end-system multicasting approach, host participating in an application session have the responsibility to forward information to other hosts depending on the role assigned by a central data and control server [15]. In this case, the architecture adopted is similar to that of IP multicast with the difference that only IP unicast service is required. End-system multicast uses an overlay structure, which is established on top of the traditional unicast services. In this way, every pair of edges (source-destination) is a unicast connection. The overlay has its meaning from the fact that the same link can have multiple unicast connections for multiple pair of edges. Although, end-system multicast seems to have many advantages (no further changes to the network are required, user has more control of the application layer, no need of special multicast router capability, etc), there is a penalty to pay. In the overlay structure, hosts are able to multicast information and consequently use the same link to redirect packets increasing the end-to-end delay of the entire transmission process [3, 8]. Another problem is the number of receivers that a potential “multicast” host can support. End users have a limited bandwidth and suffer the last mile problem [9].

While these different multicast approaches can displace some of the costs of face-to-face communications, their true potential business benefit lies in improving the accessibility and timeliness of information, vastly increasing its value to both the organization and individual employees. Although research on multicast dates back to the early days of the Internet, it has yet to produce a multicast service that is ubiquitously and economically available. In spite of the performance advantages, commercial deployment of multicast has not yet been fully realized. One of factors that prevent the wide-range deployment of multicast is the difficulty in providing reliable multicast transport.

Fortunately, recently there has been a renewed effort to realize different approaches of multicasting. In this paper, we are also going to analyze these different methods of multicasting with respect to their performance differences.

2. THEORETICAL ANALYSIS OF MULTICAST SYSTEMS

The Internet consists of interconnected LANs. A LAN may or may not have native multicasting support and the same holds for the IP layer on top of the LAN. In this section, we will theoretically analyze the problems of different level of multicasting, which hinder their performance with respect to the bandwidth utilization and latency.

2.1 Multiple Unicast Systems

In the unicast IP network, the host acting as the source transmits a copy of the message to each destination host as shown in Fig. 1. No special configuration is needed either in the source or in the core network. The intermediate routers will have to carry all these messages to the proper destinations. If, for example, the source host transmits ten copies of the same message at the

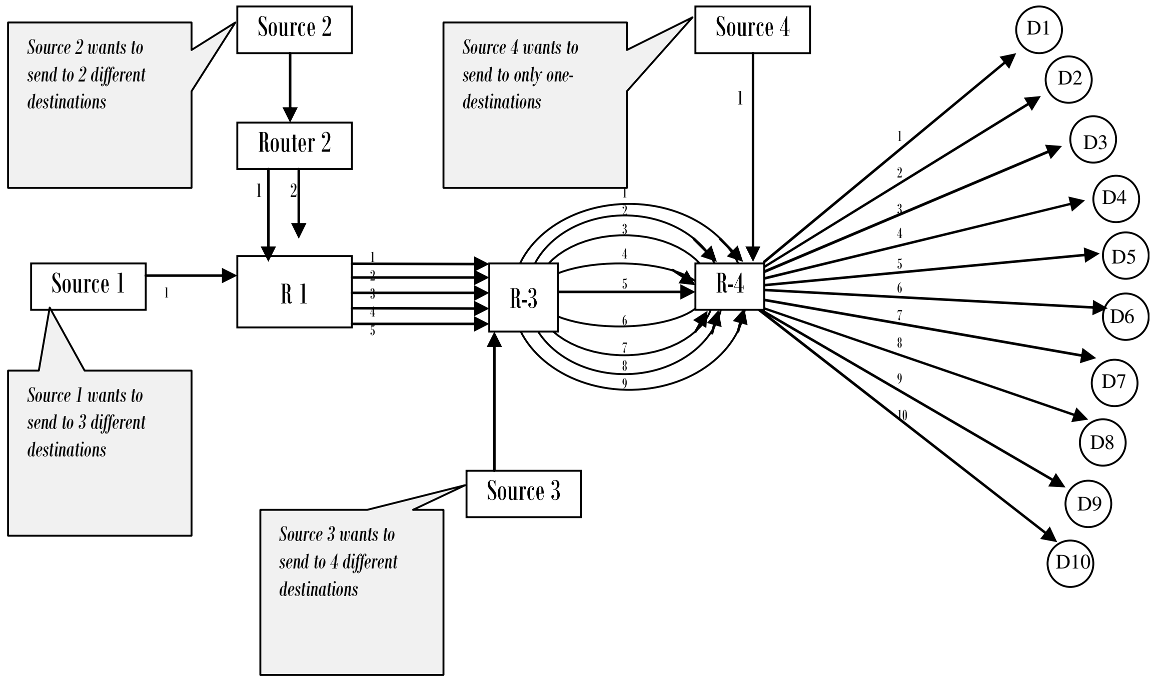
Source 2
Source 2 wants to send to 2 different destinations
Router 2
Source 4
Source 4 wants to send to only one-destinations
Source 1
R 1
R-3
R-4
Source 1 wants to send to 3 different destinations
Source 3
Source 3 wants to send to 4 different destinations
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10

FIGURE 1: Example of Multiple Unicast

same time, then obviously ten times the bandwidth of one message is required on source host's network and on the router connected to the source host's network. The chains of protocol entities that take care of the transmission process also use processing capacity on the host for each transmission. In addition, the transmission time is increased ten times and it will affect the global end-to-end delay. We pay special attention to the nearest link between the first router and the source since it is mostly there where the maximum bandwidth consumption takes place. These are the reasons to consider a multiple unicast service an unpractical approach to implement on the network.

2.2 IP Multicast Systems

IP multicast is a service where one source sends data to a group of receivers each of them containing a class D address as membership identification. IP multicast has long been regarded as the right mechanism due to its efficiency. In IP multicast, a packet is sent only once by the source. Routers along the route take care of the duplication process.

The IP-multicast capable version of the network shown in Fig. 2 consists of network with native multicast support. IP multicast capable routers are consider along the path. Efficiently routing multicast protocols are implemented. The traditional process includes the construction of a source-rooted tree together with the members of the multicast group. Since only one copy of the message is required, we can say that a minimum bandwidth effort is being used for the transmission of the message to all group members connected in the network. The problem for IP multicast is that there is no commercial support for multicast routers. Investors still think that there is not enough multicast application demand and that multicast traffic could take their routers down due to congestion problems.

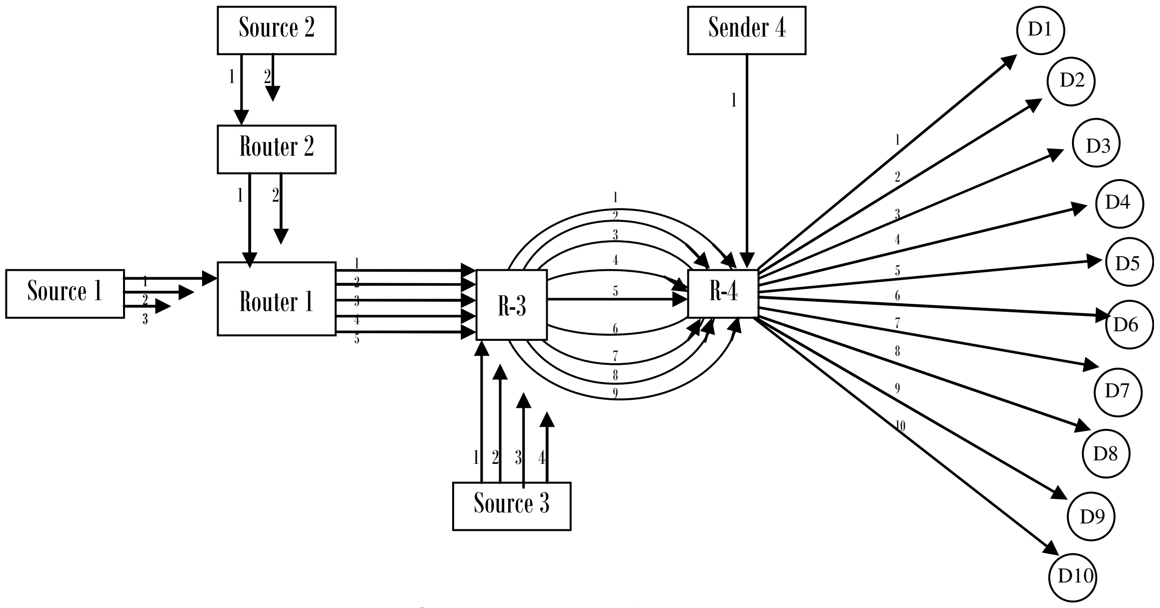

FIGURE 2: Example of IP Multicast

The IP-multicast transmission takes the same bandwidth on source host's network as a single copy, regardless of how many clients are members of the destination host group in the Internet. The obvious difference between the multiple-unicast and IP multicast is that IP multicast scales very well. Even if not all LANs have native multicast support, the added cost of transmitting copies will be limited to a single LAN.

Besides the advantages of IP multicast, there are also certain drawbacks of this approach. One of them, like we mentioned before, is the deployment problem of IP multicast, which imposes dependency on routers. The main disadvantage of IP multicast is the need of commercial routers supporting multicast protocol. In theory, almost all routers support multicast but in practice this is not the case [13]. This prevents multicast service fully implemented in Internet application (experimental research has been conducted in the Internet in a special platform named The Mbone) [10, 11].

Several approaches to multicast delivery in the network have been proposed which make some improvements or simplifications in some aspects, but they do not improve upon traditional IP multicast in terms of deployment hurdles. A major obstacle for deployment of multicast is the necessity to bridge from/to the closest multicast router to/from the end-systems. Existing IP multicast proposals [5, 6] embed an assumption of universal deployment, as all routers are assumed to be multicast capable. The lack of ubiquitous multicast support limits the deployment of multicast applications, which in turn reduces the incentive for network operators to enable multicast [12]. Therefore, from the above discussion one can expect that we need another multicast alternative in which network routers have not to do all of the work; instead each of the host will equally contribute in the overall multicast process of the messages.

2.3 End System Multicast (ESM) Systems

Because of the limitations in IP multicast, researchers have explored an alternative architecture named end-system multicast, which built a system on top of the unicast services with multicast functionalities. End-system multicast is a very promising application layer solution where all the multicast functionality is shifted to the end users as shown in Fig. 3. In this approach, host participating in an application session can have the responsibility to forward information to other

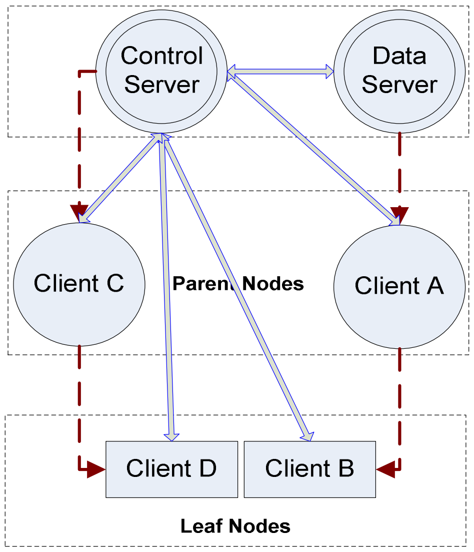

FIGURE 3: Example of ESM, Solid Lines Represent 2 Way Packet Transmission, Dotted Lines Represent One Way Packet Transmission.

hosts. Here, end users who participate in the multicast group communicate through an *overlay* structure.

However, doing multicasting at end-hosts incurs in some performance penalties. Generally, end-hosts do not handle routing information as routers do. In addition, the limitation in bandwidth and the fact that the message needs to be forwarded from host-to-host using unicast connection, and consequently incrementing the end-to-end delay of the transmission process, contribute to the price to pay for this new approach. These reasons make end-system multicast less efficient than IP multicast.

The structure of the end-system multicast is an overlay in a sense that each of the paths between the end systems corresponds to a unicast path [14]. In other words, end-system multicast is built on top of the unicast services provided by network on transport layer. Here the membership and replication functionality is performed by the end receivers, which connect together over unicast channels to form a multicast tree, rooted at one data source. The end receivers could play the role of parent or children nodes. The parent nodes perform the membership and replication process. The children nodes are receivers who are getting data directly from the parent nodes. There is one central control server and one central data server residing in the same root source. Any receiver can play the role of parent to forward data to its children. Each client has two connections: a control connection and a data connection.

3. END-TO-END DELAY APPROXIMATION FOR MULTICAST SYSTEMS

This section presents the mathematical model for approximating the end-to-end delays for all multicasting schemes. For the ease of simplicity, we divide our approximation for each type of multicasting approach such as unicast, multiple unicast, IP multicast, and ESM.

3.1 Model and Assumptions

Let G is an irregular graph that represents a network with a set of N vertices and M edges such as: $G = \{N, M\}$. Let L is a direct communication link between a single pair of source (s) and destination (*d*) where both source and destination belong to *N* such as: $\{s,d\}\in N$. In addition, each packet transmitted between source (*s*) and destination (*d*) must traverse one or more communication links in order to reach the final destination.

Let the value of *D(L)* denotes packet-delay (we sometime refer it as link delay) that is associated with each direct communication link. Therefore, each transmitted packet will typically experience a delay of *D(L)* on a particular link. The delay includes transmission, processing, and propagation delays such as: *Link-Delay = D(L) =* Transmission Delay + Propagation Delay + Processing Delay where $L \in M$. In connection less communication such as IP network, there might be multiple routes exist between a pair of source and destination. As a result, each packet might follow a different route in order to reach the final destination where each route requires traversing of one or more communication links (*L*). A single route between a pair of source and destination can be defined as: $R\{s,d\}\, where \{s,d\}\in N$. System parameters along with their definitions are presented in Table 1 and Table 2.

3.2 Mathematical Model for a Unicast System

In unicast, a packet is sent from one point (source) to another point (destination). As mentioned earlier, when packet transmit from one source (*s*) to a specified destination (*d*), there exist multiple routes where each route can have multiple links. This implies that the packet-delay for unicast is entirely dependent on the number of links a packet needs to traverse in order to reach the final destination system. Based on the above argument, one can define the packet delay such as: $D(R)=D(L_1)+D(L_2)+........+D(L_n)$ where *n* is the maximum number of links that need to be traversed on route *R* between *s* and *d*. The delay can be generalized for one particular route (*R*) that exist between source (*s*) and destination (*d*) such as:

$$D(R)=\sum_{i=1}^{n} D(L_i) \tag{1}$$

$$where \sum_{i=1}^{n}(L_i)=\{L_1+L_2+......+L_n\}\in M$$

Equation (1) can be further expressed as:

$$Delay = D_{(s-d)} = \sum\nolimits_{L\in R(s-d)} D(L) \tag{2}$$

where $L\in R(s-d)$ represents the value of the total delay associated with the route *R* between source *s* and destination *d*.

Based on (1) and (2), one can also simply derive a mathematical expression for estimating an average-delay (denoted by *AD*) which each packet may typically experience if it traverses one of the available routes. The mathematical expression for an estimated *AD* is as follows:

$$AD_{(s-d)} \square \left(\sum_{i=1}^{y} D(R_i) \Big/ y\right) \tag{3}$$

where *y* represents the maximum number of possible routes between source *s* and destination *d*.

In addition to the average delay, one can also chose the optimal route with respect to the minimum delay that each packet may experience when traverse from one particular source *s* to a destination *d* such as:

Parameters	Definition
D(L)	Denotes packet-delay that is associated with each direct communication link
s	Represents source
d	Represents destination
C_n	Represents the child node
P_n	Represents the parent node
D(R)	Denotes maximum number of links that need to be traversed on route *R* between *s* and *d*
AD	Average-delay
OD	Optimal-delay for pair of *s* and *d*
$D_{Total(MU)}$	Estimate of the total delay that the entire packet transmission will experience in a multiple unicast system
$D_{Total(M_G)}$	Approximation of total delay experienced by multicast packets when transmitted from a root node (*s*) to a multicast group (M_G)
y	Represents the maximum number of unicast routes between a source (*s*) and multiple destinations
n	Represents the maximum number of links a unicast route has
MU	Stands for multiple unicast system
M_G	Denotes a multicast group that consists of one or more destination systems
Z	Represents the size of the multicast group such as: $Z = \mid M_G \mid$
T	Represents spanning tree
D	Represents total delay experienced by multicast packets when transmitted from a root node (*s*) to a multicast group (M_G)
$L_{n,Z}$	Represents the total number of links (i.e., $Z \in R_T$) that a packet needs to traverse in order to reach the specific destination

TABLE 1: System and Parameters Definition.

$$OD_{(s-d)} = Min\left|\sum_{i=1}^{y} D\left(R_i\right)\right| \tag{4}$$

Equation (4) gives minimum delay that each packet experiences when transmitting between a pair of *s* and *d*. We refer this as an optimal delay (OD) for a pair of *s* and *d*. The above derivations can be further extended for the multiple unicast system where a single source (*s*) can transfer a packet simultaneously to multiple destinations. This hypothesis leads us to the following argument: multiple routes can be established between the source (*s*) and each destination system. The following mathematical expression can be used to estimate the total delay that the entire packet transmission will experience in a multiple unicast system:

$$D_{Total(MU)} = D_{\left\langle s \xrightarrow{multiple} \left(d_1, d_2, \ldots\ldots, d_y\right)\right\rangle} = \sum_{i=1}^{y} D\left(R_i\right) \tag{5}$$

where *y* in (5) is the maximum available unicast routes between a particular source (*s*) and multiple destinations.

Although, in multiple unicast system, a single packet can be transmitted from one source to multiple destinations, the transmitted packet may follow a different route in order to reach the appropriate destination. Consequently, each packet transmission may yield a different delay depending on the number of links the packet needs to traverse on the chosen unicast route. This leads us to the following mathematical expression for an average delay:

Parameters	Definition
ESM_G	Denotes an ESM group that consists of one or more end-system destination
X	Represents the size of the group such as $X = \mid ESM_G \mid$
N	An overlay network consists of a set of N end-system nodes connecting
R_T	Represents a route between a source node (*s*) and non-leaf nodes that could be parent or child nodes
T_T	Represents the transmission time
P_s	Represents the packet size
B_W	Denotes the bandwidth
L	Represents particular link (*L*) between a pair of source (*s*) and destination (*d*)
D_Q	Denotes queuing delay
τ	Represents the propagation delay
L_D	Denotes distance for a communication link
SoL	Represents speed of light
AB_W	Represents average-bandwidth between a pair of source (s) and destination (d)
OB_W	Represents optimal-bandwidth between a pair of source (s) and destination (d)

TABLE 2: System and Parameters Definition.

$$AD_{(s-d)} \cong \left(\sum_{i=1}^{y} \frac{D(R_i)}{y} \right) where\ D(R_i) = \sum_{i=1}^{n} D(L_i) \tag{6}$$

where *y* represents the maximum number of unicast routes between a source (*s*) and multiple destinations and *n* represents the maximum number of links a unicast route has.

3.3 Mathematical Model for a IP Multicast System

In IP multicast system, a single source (*s*) sends a packet to a group that consists of multiple destination systems. In addition, a packet is sent only once by the source system where as the intermediate routers along the route perform replications with respect to the number of destinations a group has. Let M_G denotes a multicast group that consists of one or more destination systems whereas *Z* represents the size of the group such as $Z = \mid M_G \mid$. In an IP multicast system, all multicast groups (M_G) can be typically organized in a spanning tree. We consider a spanning tree rooted at the multicast source (*s*) consisting of one of the multicast groups (M_G) that has a size of *Z*. The spanning tree can then be expressed as: $T = (N_T, M_T)$ where the numbers of destinations in one multicast group (M_G) belong to the total number of nodes present in the network such as: $M_G \in M$.

Also, Based on the above discussion, we can give the following hypothesis: The total delay (*D*) experienced by multicast packets when transmitted from a root node (*s*) to a multicast group (M_G) can be defined as a sum of the total delay experienced by each link of a spanning tree from the root nodes (*s*) to all destinations ($d \in M_G$) and the delay experienced by each link of an intermediate routers.

Thus, this leads us to the following expression for total delay (D_{Total}) experienced by multicast packets transmitted from root node (*s*) to a destination node (*d*) that exists within M_G:

$$D_{Total(M_G)} = D_{(s-M_G)} = \sum_{i=1}^{Z} D(L_i) + \sum_{i=1}^{n} D(L_i) \tag{7}$$

where *Z* is the number of destination systems in one multicast group of a spanning tree (*T*) and *n* represents the total number of links a route has.

The first term of (7) yields the total delay associated with the number of links with in a spanning tree when a packet is transmitted from a root node (source) to all the leaf and non-leaf nodes. The second term of (7) provides a total delay that a packet may experience when transmitted along a certain route.

Equation (7) can be further generalized for one of the specific destinations (*d*) within a multicast group such as $d \in M_G$, if we assume that we have a route within a spanning tree (*T*) from multicast source (*s*) to a specific destination (*d*) such as R_T (*s*, *d*), then the multicast packets transmitted from a source node to a destination experience a total delay of:

$$D_{\langle s \longrightarrow (d \in M_G) \rangle} = \sum\nolimits_{L_{n,Z} \in R_T(s,d)} D\left(L_{n,Z}\right) \tag{8}$$

where $L_{n,Z}$ represents the total number of links (i.e., $Z \in R_T$) that a packet needs to traverse in order to reach the specific destination *d* along a path of R_T with in the tree *T* as well as the number of links from source *s* to a multicast group M_G.

3.4 Mathematical Model for an End System Multicast (ESM)

Because of the limitations in IP multicast, researchers have explored an alternative architecture named ESM, which is built on top of the unicast services with multicast functionalities. In ESM, one of the end-system nodes (*s*) participating in an application session can have the responsibility to forward information to other hosts. Here, end users that participate in the ESM group communicate through an *overlay* structure. An ESM group can have at most *N* end-system nodes where we focus on one of the end-system nodes (*s*) that multicast information to the other participating nodes of a multicast end-system group. From the source host point of view, this ESM group can be considered a group of destination systems. For the sake of mathematical model, lets ESM_G denotes an ESM group that consists of one or more end-system destination where as *X* represents the size of the group such as $X = | ESM_G |$. Based on the derived expression of unicast in the previous sections, these unicast links can not exceed to *M* such as $m_1, m_2, ..., m_y \in M$ where one of the edges provides a unicast connection between two end-system nodes such as:

$$\{m \in M\} \xrightarrow{unicast-link} \{n_1, n_2\} \subset \{s, N\} \tag{9}$$

An overlay network consists of a set of *N* end-system nodes connecting though *M* number of edges where one of the end-system is designated as source host (*s*) such as: $G = \{s, N, M\}$. This also shows that an ESM is built on top of the unicast services using a multicast overlay network that can be organized in a spanning tree such as $T = (N_T, M_T)$ rooted as an ESM source (*s*) where the numbers of destinations in one multicast group (ESM_G) belong to the total number of nodes present in the network such as: $ESM_G \in M$. The end receivers in a multicast tree could be a parent or a child node depending on the location of the node.

In a multicast spanning tree (*T*), all the non-lead nodes can be both parent and child at the same time where as all the leaf nodes are considered to be the child nodes. Based on the above argument, one can say that a multicast packet originated from the root (*s*) of a spanning tree (*T*) need to traverse typically two links; source to non-leaf node (P_n, C_n) and a non-leaf node to a leaf node (C_n). Lets R_T (s, non-leaf node) represents a route between a source node (*s*) and non-leaf nodes that could be parent or child nodes such as:

$$R_T = \{P_n \vee C_n \in ESM_G\} \text{ where } P_n, C_n \in \{s \cup N\} \tag{10}$$

where, R_T (P_n, C_n) in (10) represents a route from a parent node to a child node such as: $R_T = \{P_n, C_n\} \in ESM_G$.

Equations (9) and (10) lead us to the following expression for computing the total delay involve in transmitting a multicast packet from a source node to one or more parent nodes (i.e., the delay associated with the first link of transmission):

$$D_{\left\langle s \xrightarrow{multiple-unicast} (P_n \vee C_n) \in \{s \mathrm{U} N\} \right\rangle} = \sum_{i=1}^{y} D\left(R_{i\left(s \longrightarrow P_n, C_n\right)} \right) where\ D\left(R_{ii\left(s \longrightarrow P_n, C_n\right)} \right) \mathrm{A} \sum_{i=1}^{n} D\left(L_i\right) \quad (11)$$

In (11), *y* is the maximum unicast routes between a source (*s*) and one or more non-leaf nodes and *n* represents the maximum number of links a unicast route can have. Similarly, the total delay experience by a multicast packet transmitted from a parent node to a child node can be approximated as follows:

$$D_{\left\langle P_n \xrightarrow{multiple-unicast} (C_n) \right\rangle} = \sum_{i=1}^{y} D\left(R_{i(P_n, C_n)}\right) where\ D\left(R_{i(P_n, C_n)}\right) \mathrm{A} \sum_{i=1}^{n} D\left(L_i\right) \quad (12)$$

By combining (11) and (12), the total delay experience by a multicast packet that transmitted from a source node (*s*) to a child node (*C_n*) can be approximated as:

$$D_{\left\langle s \xrightarrow{multiple-unicast} (C_n) \in \{s, P_n \mathrm{U} N\} \right\rangle} = \sum_{i=1}^{n} D\left(L_{i\left(s \longrightarrow P_n, C_n\right)} \right) + \sum_{i=1}^{n} D\left(L_{i(P_n, C_n)} \right) \quad (13)$$

4. FORMULIZATION OF BANDWIDTH EFFICIENCY FOR MULTICAST SYSTEMS

This section presents the formulization of the bandwidth efficiency for all multicasting schemes. For the ease of simplicity, we divide our proposed formulization for each type of multicasting approach such as unicast, multiple unicast, IP multicast, and ESM.

4.1 Generic System Model

In order to approximate the bandwidth for Unicast, multicast, and ESM, we use the classical definition of computing the transmission time. Based on this definition, the transmission time (T_T) can be defined as a product of the packet size (P_s) which we transmit between a pair of source (*s*) and destination (*d*) and the inverse of the bandwidth (B_W). Mathematically, this can be expressed as follows: $T_T = (P_s)\left(1/B_W \right)$

Let the value of ($D_{(s \longrightarrow d)}$) denotes the total packet-delay which is associated with each direct communication link. Therefore, each transmitted packet will typically experience a delay of $D_{(s \longrightarrow d)}$ on a particular link (*L*) between a pair of source (*s*) and destination (*d*). This delay is the sum of the transmission time, the queuing and the propagation delays such as: $D_{(s \longrightarrow d)}$ = Transmission Time (T_T) + Propagation Delay (τ) + Queuing Delay (D_Q). Also, it can be expressed as: $D_{(s \longrightarrow d)} = T_T + \tau + D_Q$. Changing the above expression for the transmission delay, we got

$$T_T = (D_{(\mathrm{s} \longrightarrow \mathrm{d})}) - (\tau) - (D_Q) \tag{14}$$

Recall our classical definition for the transmission time, (14) can be rewritten as the product of bandwidth and packet size:

$$\left(\frac{1}{B_W}\right) P_s = (D_{(\mathrm{s} \longrightarrow \mathrm{d})}) - (\tau) - (D_Q) \tag{15}$$

Solving (15) for approximating the bandwidth, we got

$$B_W = \frac{P_s}{(D_{(\mathrm{s} \longrightarrow \mathrm{d})}) - (\tau) - (D_Q)} \tag{16}$$

It should be noted that the propagation delay (τ) is a ration between the distance for a communication link (L_D) and the speed of light (*SoL*). This allows us to further extend (16) as follows.

$$B_W = \frac{P_s}{(D_{(\mathrm{s} \longrightarrow \mathrm{d})}) - \left({}^{L_D}\!/_{SoL} \right) - D_Q} \tag{17}$$

Simplifying the above equation (17), we got

$$B_W = \frac{(P_s)(SoL)}{\left| (SoL)\ (D_{(\mathrm{s} \longrightarrow \mathrm{d})})\ -\ L_D - (SoL)(D_Q) \right|} \tag{18}$$

For the sake of simplicity, we can ignore the queuing delay. Equation (18) can now be written as:

$$B_W = \frac{(P_s)(SoL)}{\left| (SoL)(D_{(\mathrm{s} \longrightarrow \mathrm{d})}) - L_D \right|} \tag{19}$$

4.2 Bandwidth Efficiency Formulization for a Unicast System

In unicast, a packet is sent from one point (source) to another point (destination). As mentioned earlier, when packet transmit from one source (*s*) to a specified destination (*d*), there exist multiple routes where each route can have multiple links. This implies that the packet-delay for unicast is entirely dependent on the number of links a packet needs to traverse in order to reach the final destination system. Based on the above argument, one can define the packet delay such as: $D(R) = D(L_1) + D(L_2) + ... + D(L_n)$ where *n* is the maximum number of links that need to be traversed on route *R* between *s* and *d*.

We generalize the delay for one particular route (*R*) that exists between source (*s*) and destination (*d*) such as: $D(R) = \sum_{i=1}^{n} D(L_i)$ where $\sum_{1}^{n} (L_i) = L_1 + L_2 + ... + L_n \in M$

This expression is further extended as: $Delay = D_{(s-d)} = \sum_{L \in R(s-d)} D(L)$ where $L \in R\left(s-d\right)$ represents the value of the total delay associated with the route *R* between source *s* and destination *d*. For a unicast system, taking the above expressions into account, the available estimated bandwidth (B_W) for a communication link (*L*) that exists between a pair of source (*s*) and destination (*d*) can be approximated in the following equation:

$$B_W = \frac{(P_s)(SoL)}{(SoL)\left\langle \sum_{L \in R\left(s \longrightarrow d\right)} D(L) \right\rangle - L_D} \tag{20}$$

The *D*(*L*) in (20) represents the link delay where as the $L \in R\left(s-d\right)$ represents the value of the total delay associated with the route *R* between source *s* and destination *d*.

The above equation (20) represents the approximated bandwidth which can be used by the transmitted packet for each individual communication link between the source and destination. It should be noted that the above equation is not representing the bandwidth approximation for one particulate route between the source and destination. Instead, it represents the bandwidth approximation for *n* number of links that need to be traversed on route *R* between source (*s*) and destination (*d*).

Based on the above derivation, one can also simply derive a mathematical expression for an average-bandwidth, denoted by AB_W. The average bandwidth represents the available bandwidth that each transmitted packet may utilize if it traverses one of the available routes. Equation (19) can be modified for the average delay between a pair of source (*s*) and destination (*d*), denoted by as follows:

$$AB_W = \frac{(P_s)(\mathrm{SoL})}{(SoL)\left\{AD_{\left(s \longrightarrow d\right)}\right\} - L_D} \tag{21}$$

The mathematical expression for an estimated AB_W can be derived as follows:

$$AB_W = \frac{(P_s)(SoL)}{\left|(SoL)\left\{\frac{\sum_{i=1}^{y} D(R_i)}{y}\right\} - L_D\right|} \tag{22}$$

Where $D(R) = \sum_{i=1}^{n} D(L_i)$ $\sum_{1}^{n} (L_i) = L_1 + L_2 + ... + L_n \in M$ and *y* represents the maximum number of possible routes between source *s* and destination *d*.

In addition to the average bandwidth, one can also choose the optimal route with respect to the minimum bandwidth that each packet may require when traverses from one particular source (*s*) to a destination (*d*). In order to derive an expression for the optimal bandwidth, we may need to modify equation (19) for the optimal delay. This is due to the fact that we assume that for each link that offers minimum bandwidth must have an optimal delay. This leads us to the following modification of (19), such as:

$$OB_W = \frac{(P_s)(SoL)}{(SoL)\ (OD_{(s \longrightarrow d)}) - L_D} \tag{23}$$

where $OD_{(s \longrightarrow d)}$ represents the optimal delay with respect to the minimum delay that each packet may experience when traverses from one particular source to a destination.

Based on (23), we can derive an expression for the optimal bandwidth, denoted by OB_W, between a pair of source (s) and destination (d) such as:

$$OB_W = \frac{(P_s)(SoL)}{\left|(SoL)\ \left\langle Min\left|\sum_{i=1}^{y} D(R_i)\right|\right\rangle - L_D\right|} \tag{24}$$

where $D(R) = D(L_1) + D(L_2) + ... + D(L_n)$ and *n* is the maximum number of links that need to be traversed on route *R* between *s* and *d*.

4.3 Bandwidth Efficiency Formulization for a Multiple Unicast System
In addition to unicast systems, we can derive the similar mathematical expressions for the multiple unicast system where a single source (*s*) can transfer a packet simultaneously to multiple destinations. In other words, in a multiple unicast system, there exist a unicast route between a source (*s*) and one of the destinations. This hypothesis leads us to the following argument: multiple routes can be established between the source (*s*) and each destination (*d_1, d_1, d_1,.......,d_y*) where *y* represents the maximum number of unicast routes established in multiple unicast. Based on this hypothesis, we can modify (19) that account the total delay such as:

$$B_{W\left\langle s \xrightarrow{multiple} (d_1, d_2, ..., d_y)\right\rangle} = \frac{(P_s)(SoL)}{\left|(SoL)\left\langle D_{s \xrightarrow{multiple} (d_1, d_2, ..., d_y)}\right\rangle - L_D\right|} \tag{25}$$

The following mathematical expression can be used to estimate the total bandwidth that the entire packet transmission utilizes in a multiple unicast system:

$$B_{WB_{W\left\langle s \xrightarrow{multiple} (d_1, d_2, ..., d_y)\right\rangle}} = \frac{(P_s)(SoL)}{\left|(SoL)\ \left\langle \sum_{i=1}^{y} D(R_i)\right\rangle - L_D\right|} \tag{26}$$

Although, in multiple unicast system, a single packet can be transmitted from one source to multiple destinations, the transmitted packet may follow a different route in order to reach the appropriate destination. In particular, a bandwidth is always associated with links rather than the complete routes between the source and destinations.

As a result, each transmitted packet may use a different amount of bandwidth with respect to the number of links that the packet needs to traverse on the chosen unicast route. This implies that, in order to estimate an average bandwidth that each packer might utilize, one should consider the number of maximum links a unicast route has. This leads us to the following mathematical expression for an average available bandwidth for the multiple unicast system:

$$AB_W = \frac{(P_s)(\mathrm{SoL})}{(SoL)\ \left\{AD_{(s \longrightarrow d)}\right\} - L_D} \tag{27}$$

Further, solving (26) for average bandwidth approximation results (27) such as:

$$AB_{W(s \longrightarrow d)}\ \mathrm{A}\ \frac{(P_s)(SoL)}{\left|(SoL)\left\langle \frac{\sum_{i=1}^{y} D(R_i)}{y} \right\rangle - L_D\right|}\ where\ D(R_i) = \sum_{i=1}^{n} D(L_i) \tag{28}$$

where y represents the maximum number of unicast routes between a source (s) and multiple destinations and n represents the maximum number of links that a unicast route has.

4.4 Bandwidth Efficiency Formulization for IP Multicast System

In IP multicast system, a single source (s) sends a packet to a group that consists of multiple destination systems. In addition, a packet is sent only once by the source system where as the intermediate routers along the route perform replications with respect to the number of destinations a group has. Lets $\boldsymbol{M_G}$ denotes a multicast group that consists of one or more destination systems whereas $\boldsymbol{Z}$ represents the size of the group such as $\boldsymbol{Z = |\ M_G\ |}$. In an IP multicast system, in order to efficiently transmit a packet from a specific multicast source to all multicast destinations, all multicast groups (M_G) can be typically organized in a spanning tree ($\boldsymbol{T}$). For the ease of mathematical expression, we only consider a spanning tree rooted at the multicast source (s) consisting of one of the multicast groups (M_G) that has a size of Z.

Based on the above discussion, we describe the spanning tree such as: $T = (N_T, M_T)$ rooted as multicast source (s) where the numbers of destinations in one multicast group (M_G) belong to the total number of nodes present in the network such as: $M_G \in M$ where M represents the total edges that the network has. The terms N_T and M_T represent the vertices and the edges of the spanning tree (T), respectively. It should be noted that we consider a spanning tree (T) that includes only the multicast destinations of a multicast group (M_G) with the exception of intermediate routers. In other words, we assume that N_T of the spanning tree (T) only consists of one or more destination nodes. The reason for this assumption is to simplify the process of estimating the total available bandwidth involves with the packet transmission in an IP multicast system.

Based on the above proposed model, we can give the following hypothesis: *The total available bandwidth (TB_W) utilizes by multicast packets when transmitted from a root node (s) to a multicast group (M_G) can be defined as a ration of packet size and the available bandwidth on each link of a spanning tree (T) from the root nodes (s) to all destinations ($d \in M_G$) with the bandwidth available to one or more intermediate routers of each link.*

The above hypothesis leads us to the following expression for total available bandwidth (TB_W) utilizes by multicast packets transmitted from root node (s) to a destination node (d):

$$TB_W = \frac{(P_s)(SoL)}{(SoL)\ (D_{(\mathrm{s} \longrightarrow M_G)}) - L_D} \tag{29}$$

Computing the total delay for the IP multicast and using its resulting values in (29), we provide an expression for the total available bandwidth such as:

$$TB_W = \frac{P_s}{\left| \left\langle \sum_{i=1}^{Z} D(L_i) + \sum_{i=1}^{n} D(L_i) \right\rangle - \frac{L_D}{SoL} \right|} \tag{30}$$

where *Z* in the denominator of (30) represents the number of destination systems in one multicast group of a spanning tree (*T*) where *n* represents the total number of links a route has.

The first term of the denominator of (30) ($\sum_{i=1}^{Z} D(L_i)$)yields the total delay associated with the number of links within a spanning tree when a packet is transmitted from a root node (source) to all the leaf and non-leaf nodes (i.e., the multiple receivers with in *T* excluding the source node (*s*)). In other words, according to spanning tree, if a message is transmitted from a source node (*s*) to all the destination nodes, then the packet must traverse *Z* (i.e., the number of destinations in one multicast group) number of links which consequently experience a delay on each link.

The second main term of the denominator ($\sum_{i=1}^{n} D(L_i)$) in (30) provides a total delay that a packet may experience when transmitted along a certain route (i.e., from a source node (*s*) to multicast group (M_G) via one or more routers along the route). The last term of the denominator can not be considered as a design parameter and therefore its value completely depends on the type of network.

The above equation can be further generalized for one of the specific destinations (*d*) within a multicast group such as $d \in M_G$, if we assume that we have a route within a spanning tree (*T*) from multicast source (*s*) to a specific destination (*d*) such as $R_T(s, d)$, then the multicast packets transmitted from a source node may utilize a total available bandwidth such as:

$$TB_{W\left\langle s \longrightarrow (d \in M_G) \right\rangle} = \frac{P_s}{D_{\left\langle s \longrightarrow (d \in M_G) \right\rangle} - \left\langle L_D \big/ (SoL) \right\rangle} \tag{31}$$

Determine the total delay for the multicast packets that transmit from a source node to one of destinations within a group and using the resulting expression in (31), we got,

$$TB_{W\left\langle s \longrightarrow (d \in M_G) \right\rangle} = \frac{(P_s)(SoL)}{\left| (SoL) \left\langle \sum_{L_{n,Z} \in R_T(s,d)} D(L_{n,Z}) \right\rangle - L_D \right|} \tag{32}$$

where $L_{n,Z}$ in (32) represents the total number of links (i.e., $Z \in R_T$) that a packet needs to traverse in order to reach the specific destination *d* along a path of R_T within the tree *T* as well as the number of links from source *s* to a multicast group M_G.

Since only one copy of the message is required in IP multicast, we can say that a minimum bandwidth effort is being used for the transmission of the message to all group members connected in the network. This minimum bandwidth is achieved due to the fact that a minimal transmission time is required for the IP multicast which in turns reduces the overall end-to-end delay as can be seen in the denominator of (32).

4.5 Bandwidth Efficiency Formulization for End System Multicast (ESM) System

An ESM group can have at most *N* end-system nodes where we focus on one of the end-system nodes (*s*) that multicast information to the other participating nodes of a multicast end-system group. From the source host point of view, this ESM multicast group can be considered a group of destination systems. For the sake of mathematical model, lets ESM_G denotes an ESM group that consists of one or more end-system-destinations where as *X* represents the size of the group such as $X = | ESM_G |$.

In an overlay network, all participating end-system nodes are fully connected to each other via the unicast links. Based on the derived expression of unicast in the previous sections, these unicast links that provide connection between end-system nodes can not exceed to *M* such as: $\{m_1, m_2, ..., m_y\} \in M$ where one of the edges (m) provides a unicast connection between the two end-system nodes such as: $\{m \in M\} \xrightarrow{unicast-link} \{n_1, n_2\} \subset \{s, N\}$.

The structure of the ESM is an overlay network in a sense that each of the paths between the end-systems corresponds to a unicast path. This implies that an overlay network consisting of a set of *N* end-system nodes connecting though *M* number of edges where one of the end-system is designated as a source host (*s*) can be expressed as: $G = \{s, N, M\}$.

This also shows that an ESM is built on top of the unicast services using a multicast overlay network that can be organized in a spanning tree such as $T = (N_T, M_T)$ rooted as an ESM source (*s*) where the numbers of destinations in one multicast group (ESM_G) belong to the total number of nodes present in the network such as: $ESM_G \in M$. Here, the membership and replication functionality is performed by the ESM receivers, which connect together over unicast channels to form a multicast tree (*T*), rooted at one data source (*s*). The end receivers (i.e., the number of end-systems in ESM_G) in a multicast tree could be a parent or a child node depending on the location of the node. In a multicast spanning tree (*T*), all the non-leaf nodes can be both parent and child at the same time where as all the leaf nodes are considered to be the child nodes. In other words, the parent nodes perform the membership and replication process where as the children nodes are receivers that receives multicast packet from the parent nodes.

Based on the above argument, one can say that a multicast packet originated from the root (*s*) of a spanning tree (*T*) need to traverse typically two links; source to non-leaf node (P_n, C_n) and a non-leaf node to a leaf node (C_n). Lets R_T (s, non-leaf node) represents a route between a source node (*s*) and non-leaf nodes that could be parent or child nodes such as: $R_T = \{P_n \vee C_n \in ESM_G\}$ where $P_n, C_n \in \{s \cup N\}$

Similarly, R_T (P_n, C_n) represents a route from a parent node to a child node such as: $R_T = \{P_n, C_n\} \in ESM_G$. The above arguments lead us to the following expression for computing the total bandwidth available for transmitting a multicast packet from a source node to one or more parent nodes (i.e., the bandwidth associated with the first link of transmission):

$$TB_{W\left\langle s \xrightarrow{multiple-unicast} (P_n \vee C_n) \in \{s \cup N\}\right\rangle} = \frac{P_s}{\left| D_{\left\langle s \xrightarrow{multiple-unicast} (P_n \vee C_n) \in \{s \cup N\}\right\rangle} - \left(L_D \middle/ SoL \right) \right|} \quad (33)$$

Determining the mathematical expression for the total delay involve in transmitting a multicast packet from a source node to one or more parent nodes and using the resulting expression in (33) yields (34) for approximating the total bandwidth such as:

$$TB_{W\left\langle s\xrightarrow{multiple-unicast}(P_n \vee C_n)\in\{s\mathrm{U}N\}\right\rangle} = \frac{P_s}{\left|\sum_{i=1}^{y} D\left(R_{i\left(s\longrightarrow P_n,C_n\right)}\right) - \left(L_D \big/ SoL\right)\right|} \quad (34)$$

$$where\ D\left(R_{ii\left(s\longrightarrow P_n,C_n\right)}\right) \mathrm{A} \sum_{i=1}^{n} D\left(L_i\right)$$

where *y* in (34) represents the maximum unicast routes between a source (*s*) and one or more non-leaf nodes and *n* represents the maximum number of links a unicast route can have.

Based on (34), we can derive a similar mathematical expression for approximating the total bandwidth available for a multicast packet which is transmitted from a parent node to a child node as follows:

$$TB_{W\left\langle P_n\xrightarrow{multiple-unicast}(C_n)\right\rangle} = \frac{P_s}{\left|\sum_{i=1}^{y} D\left(R_{i(P_n,C_n)}\right) - \left(L_D \big/ SoL\right)\right|}\ where\ D\left(R_{i(P_n,C_n)}\right) \mathrm{A} \sum_{i=1}^{n} D\left(L_i\right) \quad (35)$$

When comparing (34) with (35), it can be clearly evident that only the total delay component in the denominator of both equations differ with each other for both presented scenarios which in turns change the amount of available total bandwidth. By combining the above two equations, the total bandwidth that a multicast packet may utilize when transmitted from a source node (*s*) to a child node (C_n) can be approximated as follows:

$$TB_{W\left\langle s\xrightarrow{multiple-unicast}(C_n)\in\{s,P_n\mathrm{U}N\}\right\rangle} = \frac{(P_s)(SoL)}{\left|\left\langle \sum_{i=1}^{n} D\left(L_{i\left(s\longrightarrow P_n,C_n\right)}\right) + \sum_{i=1}^{n} D\left(L_{i(P_n,C_n)}\right)\right\rangle (SoL) - L_D\right|} \quad (36)$$

As one can see in (36) that the hosts are able to multicast information and consequently use the same link to redirect the packets resulting in an increase in the end-to-end delay of the entire transmission process. The limitation in bandwidth and the fact that the message needs to be forwarded from host-to-host using unicast connection, and consequently incrementing the end-to-end delay of the transmission process as can be seen in the denominator of (36), contribute to the price to pay for this new approach. For the ESM, this price will be paid in terms of the second quantity of the denominator of (36). (i.e., $\sum_{i=1}^{n} D\left(L_{i(P_n,C_n)}\right)$). These reasons might make ESM slightly less efficient than the IP multicast system.

5. PERFORMANCE ANALYSIS AND EXPERIMENTAL VERIFICATIONS

In this section, we provide the simulation results for analyzing the performance of different multicast schemes with respect to the delay analysis and the bandwidth approximation.

5.1 System Model

Simulations are performed using OPNET to examine the performance of Multiple unicast, IP multicast, and ESM schemes. Figure 4 shows an OPNET model for the multiple unicast, IP multicast and ESM simulations. The OPNET simulation has run for a period of 900 seconds for all three scenarios where we collect the simulated data typically after each 300 seconds. For all scenarios, we have setup one sender node that transmits video conferencing data at the rate of

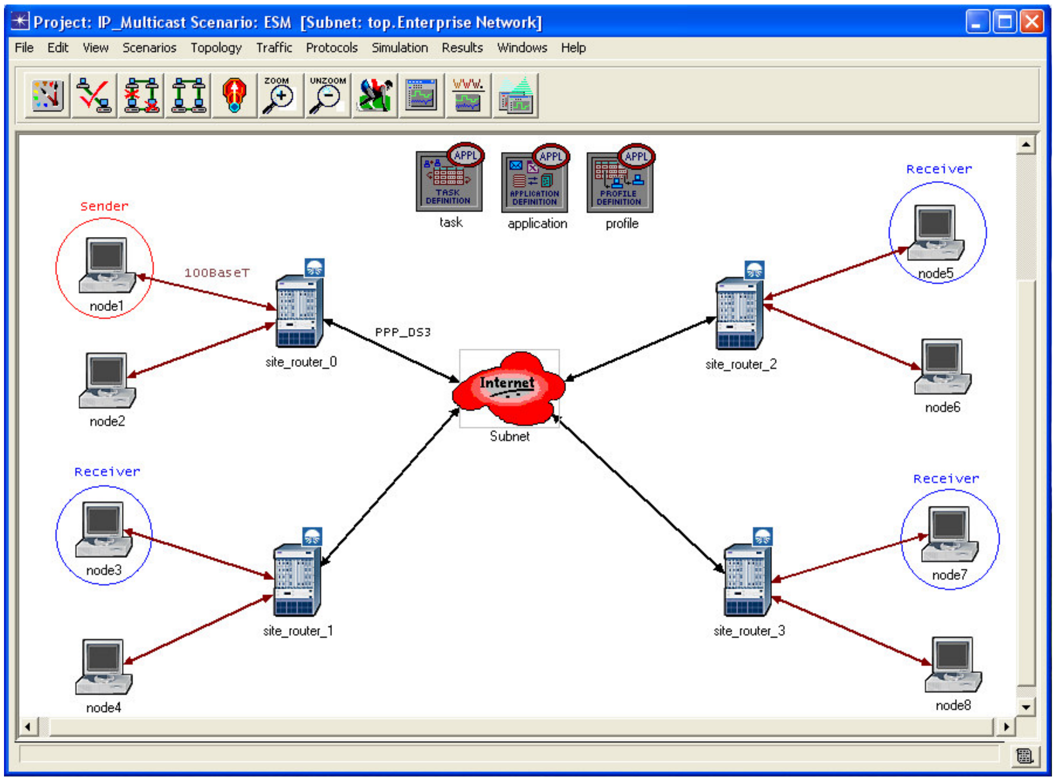

Figure 4: OPNET Model for Multiple Unicast, IP Multicast and End-System Multicast Video Conferencing Transmissions.

10 frames/s using 2,500-stream packet size to one or more potential receivers via a link that operates at 100 Mbps. In addition to these 100 Mbps licks, we use separate DS3 links for the core network (Internet). The same traffic pattern is assumed for all scenarios.

It should be noted in Fig. 4 that we use four backbone routers that connect multiple subnets to represent Bay Networks concentrator topology using ATM – Ethernet FDDI technology. In order to generate consistent simulation results, we use the same topology for the first two scenarios with some minor exceptions. For Multiple unicast, we disable the multicast capabilities of backbone routers where as for the IP multicast this restriction does not impose. Finally, in order to examine the behavior of the ESM, we use an OPNET Custom Application tool that generates the overlay links and the source root.

5.2 Experimental Verifications

For the Multiple unicast scenarios, video conferencing data is being sent by the root sender at the rate of 25 K-bytes per second. This implies that a total of three copies traveling which result in 75 K-bytes per second of total traffic. The last mile bandwidth limitation typically provides the most important delay impact. OPNET collected all the delays for all the receivers and calculated the average. The packet end-to-end delay for multiple unicast was 0.0202 seconds. For the IP multicast approach only *one copy* of the packet is generated at the root source. For this reason, the total video-conferencing traffic sent and received is only 25,000 bytes/s. Thus, a better performance in the average packet end-to-end delay can be observed. This is approximately 0.0171 seconds. Finally, after performing ESM simulations, we obtain an average end-to-end delay packet of 0.0177 seconds.

It can be seen in Fig. 5 that ESM packets transmission provides comparatively good performance than the Unicast but not as good as the IP multicast. The reasons are the RDP (Relative Delay

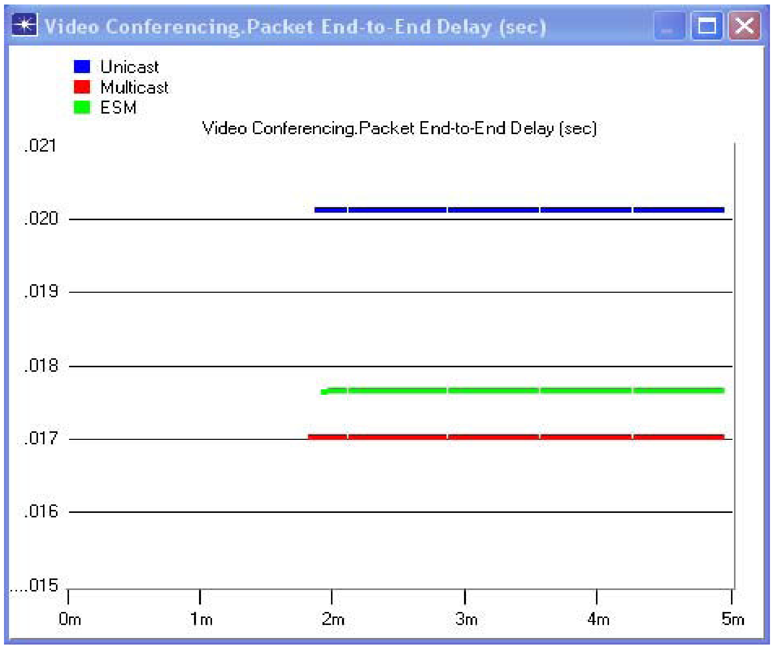

Figure 5: Average end-to-end packet delays for multiple unicast, IP multicast and ESM simulations.

Penalty or the ratio of the delay between the sender and the receiver) [6] and the LS (Link Stress or the number of identical copies of a packet carried by a physical link) experienced by each network schemes. Even though, a Unicast scheme provides comparatively low RDP, the value of LS is not optimal.

On the other hand, IP multicast performs with a little bit higher RDP but it gets a better LS. ESM has the inconvenience of RDP higher than IP multicast due to the fact that for a second receiver, there is an increasing delivery–delay because of the end-user replication (the second user has to wait for the data sent by its father node or sub-server). This is the penalty that ESM has to pay. One possible solution would be the design of a robust multicast protocol to optimize the delivery of data for the final users. Note that the additional delay could be reduced if we optimize the bandwidth utilization in the potential parent nodes. This is not a simple task because it requires a smart protocol to recognize bandwidth limitations in potential parent nodes and to establish an algorithm to limit the number of children nodes for these parent nodes.

6. CONSLUSION & FUTURE WORK

In this paper, we presented both analytical and mathematical models for all the multicast approaches currently available for multimedia applications. We first presented a mathematical model for multiple unicast systems in which the source host has to send a single copy of data to every single receiver. Our proposed formulization suggested that this approach wastes a lot of significant network bandwidth. Secondly, we presented a mathematical model for the IP multicasting approach which is a more efficient concept where the data source only sends one copy of data which is replicated as necessary when propagating over the network towards the receivers. Our proposed formulization shows that the IP multicast demonstrates some good bandwidth efficiency characteristics than the other multicast approaches. Finally, we presented a complete formulization of bandwidth efficiency for ESM systems, which is an alternate to router-dependent multicast service that allows end-systems participating in a multicast group to replicate and forward packets to other receivers. Our proposed formulization of bandwidth efficiency suggests that the ESM is a feasible, especially for sparse, medium size group. The simulation results presented in this paper fully support the proposed analytical model for the IP multicast and ESM.

7. REFERENCES

1. Y. Ma, R. Chbeir, and K. Yetongnon. "*An improved cluster-based P2P multicasting approach*". Proceedings of the 2nd international conference on Scalable information systems, Vol. 304, No. 59, June 2007.

2. D. Kostić, A. Snoeren, A. Vahdat, R. Braud, C. Killian, J. Anderson, J. Albrecht, A. Rodriguez, E. Vandekieft. "*High-bandwidth data dissemination for large-scale distributed systems*". Transactions on Computer Systems (TOCS)**,** 26(1), February 2008.

3. S. Fahmy and M. Kwon. "*Characterizing overlay multicast networks and their costs.*" IEEE/ACM Transactions on Networking (TON)*,* 15(2):373 – 386, April 2007.

4. S. Rizvi, A. Riasat, and M. Elleithy. "*Deterministic formulization of end-to-end delay for multicast systems.*" Accepted in International journal of Computer Networks & Communications, Vol. 2, No.1, 2010.

5. H. Holbroo and D. Cheriton. "*IP multicast channels: EXPRESS support for large-scale single-source applications*." In Proceedings of SIGCOMM, 1999.

6. B. Fenner, M. Handley, H. Holbrook and I. Kouvelas. "Protocol independent multicast-sparse mode (PIM-SM): protocol specification (Revised)." IETF Internet Draft, work in progress, March 2003, draft-ietf-pim-sm-v2-new-07.txt

7. D. Kosiur. "*IP Multicasting: The complete guide to interactive corporate networks.*" New York: John Wiley & Sons, Inc., 1998.

8. J. Saltzer, D. Reed, and D. Clark. "*End-to-End arguments in system design*." ACM Transactions on Computer Systems*,* 2(4):195–206, 1984.

9. Y. Chua, S.Rao, S.Sechan and H. Zhang. "*Enabling conferencing applications on the internet using an overlay multicast architecture.*" Proceedings of the ACM SIGCOMM'01, San Diego, CA, August 2001.

10. S. Deering. "*Multicast routing in inter-networks and extended LANs.*" Proceedings of the ACM SIGCOMM 88, pp. 55–64, Stanford, CA, August 1988.

11. Y. Chu, S. Rao, and H. Zhang. "*A case for end system multicast*." Proceedings of ACM Sigmetrics, June 2000.

12. S. Y. Shi and J.S. Turner. "*Multicast routing and bandwidth dimensioning in overlay networks*." IEEE Journal on Selected Areas in Communications (JSAC), 20(8):1444–1455, October 2002.

13. S. Shi. "*Design of overlay networks for Internet multicast.*" *PhD dissertation, Washington University in St. Louis*, August 2002.

14. M. Blumenthal and D. Clark. "*Rethinking the design of the Internet: The end-to-end argument vs. the brave new world*." ACM Transactions on Internet Technology, 1(1): 70–109, August 2001.

15. M. Kwon and S. Fahmy. "*Topology-aware overlay networks for group communication.*" Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video, pp. 127 – 136, May 2002.

Soft Real-Time Guarantee for Control Applications Using Both Measurement and Analytical Techniques

Baek-Young Choi choiby@umkc.edu
University of Missouri,
Kansas City, MO, 64110, USA

Sejun Song sjsong@tamu.edu
Texas A&M University,
College Station, TX, 77843, USA

Abstract

In this paper, we present a probabilistic admission control algorithm over switched Ethernet to support *soft real-time* control applications with *heterogeneous periodic* flows. Our approach is purely end host based, and it enables real-time *application-to-application* QoS management over switched Ethernet without sophisticated packet scheduling or resource reservation mechanisms in Ethernet switches or middleware on end hosts. In designing the probabilistic admission control algorithm, we employ both measurement and analytical techniques. In particular, we provide a new and efficient method to identify and estimate the queueing delay probability inside Ethernet switches for heterogeneous periodic flows with variable message sizes and periods. We implemented the probabilistic admission control algorithm on the Windows operating system, and validated its efficacy through extensive experiments.

Keywords: soft real-time, probabilistic admission control, periodic flows.

1. INTRODUCTION

A typical mission-critical real-time control system consists of sensors, actuators, controllers, data-intensive devices, and instrumentation devices. It works with the combination of periodic closed control loops. Controllers receive inputs from various sensors and data-intensive devices and perform control logic that determines how actuators and instrumentation devices should be operated. Each control loop has its Quality-of-Service (QoS) requirements, in particular, timing requirements including delay, jitter, and loss for a certain size message transmission. Traditionally, the mission-critical real-time control systems are designed to support the *hard* guarantee of their QoS requirements. However, for many practical real-time control systems found in industrial process controls, real-time signal processing, and telecommunications, a *hard* guarantee could be considered overly stringent by requiring excessive system resources. The *statistical* or *soft* guarantee accepts the performance as long as the violation probability of QoS requirements is below the pre-specified level. It is desirable to be designed and utilized for many real-time control applications to allow an efficient resource usage. In this paper, we focus on real-time control systems with soft QoS requirements, more specifically, soft delay guarantees.

Various real-time control networks have been developed with proprietary hardware and protocol solutions to provide deterministic controls for the specific applications. However, recent trends in the mission-critical control system industry replace proprietary networks with commercial-off-theshelf (COTS) or open networks so as to reduce product development cycle time and cost as well as to achieve system interoperability. Moreover, high-bandwidth sensors (e.g., infrared video, acoustic, and color sensors) are becoming increasingly common in control networks. Due to its ubiquity, simplicity and low cost, Ethernet has become a de facto choice for developing open mission-critical network strategies [2], [20], [13]. However, as the traditional Ethernet is a shared communication network using the CSMA/CD (Carrier Sense Media Access/Collision Detect) MAC protocol, packets to be transmitted may be held back arbitrarily long due to the random exponential back-off algorithm used to avoid collision. In other words, packet transmission delay over the traditional Ethernet is unpredictable, which makes it difficult to build a real-time control network over the traditional Ethernet. A better way to build real-time control systems over a local area network (LAN) is to use the switched Ethernet technology. Because of its switching capacity, an Ethernet switch not only enables fast packet transmission, but also reduces the chance of packet collision. Furthermore, with its internal buffer, a switched Ethernet can temporarily buffer packets that are competing for the same output port, further reducing the chance of an inside switch packet collision. However, a shared buffer inside of Ethernet switches introduces variable queueing delays of the packet transmission [16]. To build real-time control systems over switched Ethernet, it needs additional mechanisms to orchestrate competing resources according to the QoS requirements. However, QoS aware reservations or admission mechanisms are not available at Ethernet switches, unlike IP based solutions on more sophisticated IP routers.

In this paper, we propose a novel *probabilistic* admission control approach on the switched Ethernet environments to enable soft real-time guarantees for the mission-critical control applications. The proposed approach is designed for the typical control applications, which have *various periodic packet flows with different packet sizes*. It performs an admission control on the end control host, that starts a periodic real-time control flow, to determine whether a new connection (or a *flow*) between two control end hosts can be established. This admissibility check ensures that the *application-to-application* delay requirements of the new and existing flows can be guaranteed within a pre-specified probabilistic delay bound. The probabilistic admission control algorithm on the end control host works as follows. It first measures the *baseline* delay distribution when there are no competing flows in the switched Ethernet control network. This initial measurement of the baseline delay distribution captures the “fixed” delay components, including propagation delay, packet transmission time, and operating system overhead. According to the baseline delay distribution and the information regarding flow requests, it then estimates the probability of a queueing delay at the Ethernet switches when there are multiple competing flows for the same output port. It provides an efficient method to estimate the probability of a queueing delay for heterogeneous periodic flows in order to obtain a probabilistic delay bound.

The proposed approach is a pure end-host based solution that does not require any software or hardware modification to the Ethernet switches. For easy deployment, it is designed in the application layer that does not require any sophisticated middleware installation in the end host Operating System kernel. We implement the admission control algorithm on the COTS OS based end-hosts and conduct extensive experiments over the switched Ethernet environments. Through the experiments, we validate the effectiveness of the proposed probabilistic admission control approach.

The remainder of the paper is organized as follows. We first discuss the related work on real-time scheduling and admission control in a LAN control network environment in Section 2. In Section 3, we describe the problem setting, the queueing analysis and the proposed admission control algorithm in detail. The software design and implementation is presented in Section 4. In Section 5, we describe the experiment design and results. The paper is concluded with a summary of the work and future research directions in Section 6.

2. RELATED WORK

In the area of real-time research, a lot of efforts have been made to provide hard deadline guarantees over Ethernet with the expense of resource utilization and average performance. Most of the earlier work [21], [17], [7] modified the Ethernet MAC sub-layer to achieve a bounded channel access time. These proprietary approaches are quite costly compared to using the well-established and widely-used current Ethernet standard. Both [24], [6], [15] proposed a virtual non-collision token ring implementation over the collision-based Ethernet. Since the token management protocol is executed by the higher-layer (OS kernel of the hosts) rather than the MAC, the approach does not need to modify the network hardware architecture. The major shortcomings of this approach are the overhead of heavy token management including the token relay among the hosts and the restoration of the lost token and the performance limitation due to the overly conservative network usage. Recent studies try to achieve the hard real-time guarantee without modifying network hardware architecture. [12] proposed a traffic shaping software on the Ethernet switch to achieve hard guarantees with bounded delays and reserved bandwidths. The proposed solution in [8] designed on standard Ethernet switches with Layer 2 QoS/CoS protocol for traffic prioritization (IEEE 802.1p). It requires a separate queue for each priority class on the switch, but it cannot be smoothly deployed with currently widespread Ethernet switches that have a common buffer to share all priority classes. It may lead to the miss of QoS guarantees for high priority packets; for example, if the common buffer is already packed by the lower priority packets. [9] tried to resolve the problem of the shared buffer by using additional traffic shaping mechanisms available in other higher layer network elements such as routers.

While there is a lot of research on designing, validating, and facilitating traditional hard real-time systems, only few such techniques exist on soft real-time systems in spite of the recent proliferation of soft real-time applications. The existing soft real-time research mostly is focused on schedulability analysis techniques. Probabilistic Time Demand Analysis (PTDA) [23] and Statistical Rate Monotonic Scheduling (SRMS) [3] are the algorithms about the statistical behavior of periodic tasks to facilitate better design of soft real-time systems.

PTDA attempts to provide a lower bound of the missing deadline probability that is determined by the time supply that equals or exceeds the time demand at the deadline of the task. The time demand is computed by convolving the probability density functions of the execution times. It assumes that the relative deadline of all tasks are less than or equal to their period. SRMS attempts to schedule tasks with highly variable execution times in such a way that the portion of the processor time allocated to each task is met on the average. Variable execution times are smoothed by aggregating the executions of several jobs in a task and allocating an execution time budget for the aggregate. A job is released only if its task contains a sufficient budget to complete it in time and if higher priority jobs will not prevent its timely completion.

In a statistical real-time guarantee work [10], they analyzed the Ethernet MAC protocol using a semi-Markov process model and derived a network-wide input limit for achieving a target transmission success ratio. The network-wide input limit is kept by enforcing each component station to control its instantaneous traffic arrival rate under its station-wide input limit. To this end, they implemented a traffic smoothing middleware between the transport layer and the Ethernet data link layer at each station. The middleware keeps the traffic arrival rate under the station-wide input limit by smoothing a busty packet stream. An enhancement on the traffic smoother is made by [5]. They used the overall throughput in tandem with the number of collisions as network load indicators to feed into their fuzzy traffic smoother to give the flexibility on the sporadic traffic process. Unlike our approach based on switched Ethernet, these studies focused on designing a traffic smoother using an ordinary shared Ethernet hub.

Queueing system studies on multiplexing periodic flows have been limited to flows with the same packet size [14] ($N * D/D/1$, $\sum D_i /D/1$ queue) in the past. A relevant work in the switch can be found in Raha et al.'s work on real-time ATM [19], [18]. Their research focuses on the worst case queuing analysis. They devised Gamma functions to represent the flows' worst-case bandwidth requirements across different time scales. These gamma functions can be used to compute the worst-case queueing delay in an ATM switch buffer. Since ATM cells are of fixed size, the switch's processing rate is constant. Therefore, the worst-case delay on each node can be computed. Since the exact gamma functions are too complex to compute, they designed three approximation methods. However, their analysis based on gamma functions only works for fixed packet size such as ATM cells. It is therefore not applicable for switched Ethernet with variable packet size.

3. PROBABILISTIC ADMISSION CONTROL APPROACH

3.1 Problem Setting

Although the proposed probabilistic admission control approach can be implemented in either a distributed or a centralized fashion, figure 1 illustrates a distributed approach based control network environment. The control network environment has several control application hosts connected by a typical Ethernet switch with simple FIFO port buffers. The probabilistic admission control software is distributed to each control host running on the application layer. Each control host also maintains the control flow information database that is synchronized over the entire control network. *Flow* is used to refer to a connection between two control applications to transmit periodic messages. When a flow is requested on a control host, the host performs an *admissibility test* to check whether its delay requirement can be satisfied within the pre-specified probabilistic delay bound. If the flow request is admitted, a broadcast message of flow addition is sent to other control hosts in order to update their flow information database. When a flow is terminated, a broadcast message of flow deletion is sent to remove the flow from the distributed hosts' flow information database. In general, since a distributed approach has multiple admission control points (on each control host) over the control system network, the concurrent flow requests on the distributed control hosts can potentially lead to unexpected QoS violations. To deal with this issue, there is a proposal [11] to utilize a safety margin in order to absorb the potential impact of concurrency.

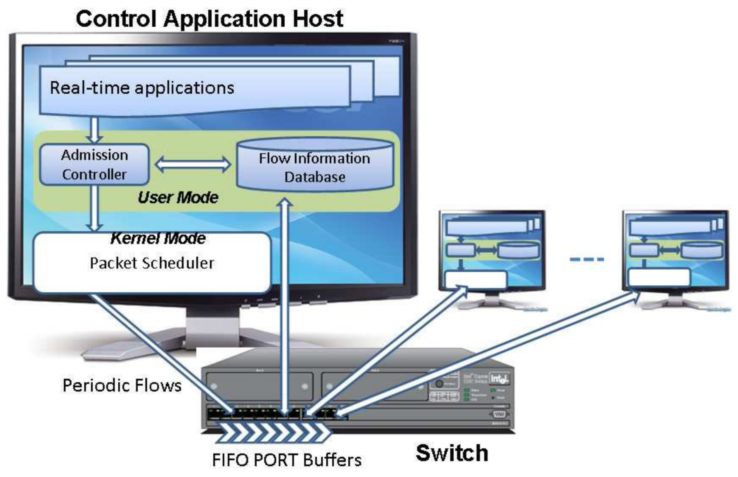

FIGURE 1: Network System Architecture.

The admission control algorithm is designed to perform an efficient on-line admissibility test by simplifying the calculation formula of the delay probability estimation. We consider the periodic flow characteristics with various periods and different message sizes. As summarized in Table I, a flow with its QoS requirements is defined by S_i (L_i, P_i, D_i, DP_i) where L_i is the (maximum) message size (bytes), P_i is the period of the flow (ms), D_i is the maximum acceptable delay (µs), and DP_i is the bound on the probability that the actual message delay *d* exceeds D_i (i.e., $P\,r(d \geq D_i) \leq DP_i$). We assume that all flows are independent of each other.

Notation	Explanation
S_i (L_i, P_i, D_i, DP_i)	a flow with:
L_i	message size (bytes)
P_i	period (ms)
D_i	maximum acceptable delay (µs)
DP_i	minimum acceptable probability that the actual delay is less than D_i (i.e., $\Pr(d \leq D_i) \geq DP_i$)

TABLE 1: Flow Specification.

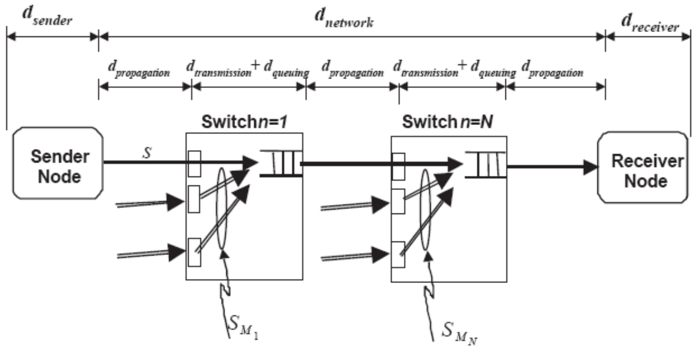

FIGURE 2: Network Delay Model.

As illustrated in figure 2, the performance metric (i.e., the message delay *d*) is the *application-to-application* delay that consists of the processing delays on both hosts and a network transmission delay as shown below:

$$d = d_{sender} + d_{network} + d_{receiver}. \tag{1}$$

The delay in network $d_{network}$ can be further decomposed into three components:

$$d_{network} = d_{\text{propagation}} + d_{\text{transmission}} + d_{\text{queueing}}, \tag{2}$$

where $d_{\text{propagation}}$ is the propagation delay of a message across the network, $d_{\text{transmission}}$ is the transmission time of a message, and d_{queueing} is the queueing delay experienced by a message at Ethernet switches. The main variability in application-to-application delay is contributed by d_{queueing} that varies with the network load and the number of competing flows in the network. The other delay components are independent of any competing flows in the network system. Hence, we group these delay components together and denote it as the baseline delay, d_{baseline}, i.e.,

$$d_{\text{baseline}} = d_{sender} + d_{receiver} + d_{\text{propagation}} + d_{\text{transmission}}. \tag{3}$$

This baseline delay of a flow can be estimated by directly measuring application-to-application delay with no competing flows in switches along the transmission path. In general, the measured baseline delay of a flow demonstrates certain distribution patterns, rather than a single delay point, due to the inherent delay variations introduced by the host OS and possible measurement errors. In figure 4, we illustrate a hypothetical probability density function, *f(x)*, of the baseline delay $d_{baseline}$.

Given the measured baseline delay distribution, in the remainder of this section we devise an efficient method to estimate the queueing delay probability when there are competing flows at a switch. In section 3.2, we consider the simple case of two competing flows. The general case with multiple competing flows is then analyzed in section 3.3. Based on the analysis, we devise an efficient probabilistic admission control algorithm that is presented in section 3.4.

3.2 Simple Case: Two Competing Flows

In this section we consider the simple case where there are only two flows competing for the same output port. Note that even when the network load is relatively light-loaded, packets may still be queued at a switch, due to the coincidental arrival of packets forwarded to the same output port. In the following, we analyze the probability that packets will be queued at a switch under the assumption that there are only two competing two flows.

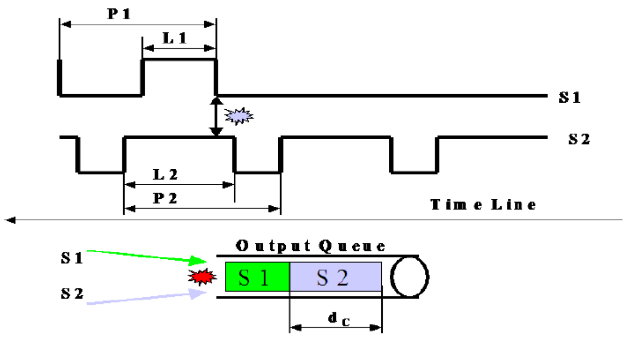

FIGURE 3: An Example of Competing Flows.

Suppose two flows, flows S_1 (L_1, P_1, D_1, DP_1) and S_2 (L_2, P_2, D_2, DP_2) send periodic transmissions through the same switch output port and the queue is initially empty. As illustrated in figure 3, if a packet of S_1 arrives before an S_2 packet is transmitted, S_1's packet would be delayed by S_2's packet. We assume that only one S_2 packet is in the queue when an S_1 packet arrives. This assumption is reasonable in a practical system environment and we will discuss in detail later. The worst case queueing delay experienced by an S_1 packet will be the time it takes the switch to transmit a whole S_2 packet. Since it depends on the message size and the period of competing flow S_2, we can calculate the queueing probability p of the flow S_1 as follows:

$$p = \frac{T2}{P2} \qquad \text{where } T2 = \frac{L2}{Switch_{Capacity}} \tag{4}$$

We denote the corresponding queueing delay T_2 that is the transmission time of L_2, as d_C for consistency with the next section.

A packet of S_1 will be delayed by d_C with a probability of *p*. There will be no queueing delay for the packet with a probability of 1 – *p*. Hence, the estimated delay for flow S_1 with the presence of competing flow S_2 is

$$d = (d_{baseline} + d_{queueing}) * p + d_{baseline} * (1 - p) \tag{5}$$

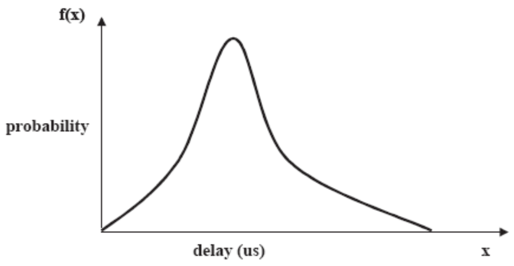

FIGURE 4: Probability Density Function for Baseline Delay.

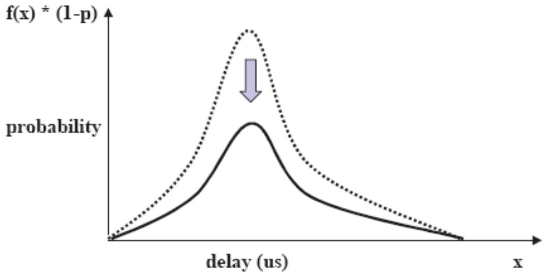

FIGURE 5: Partial Probability Density Function for No-Queuing Case.

As we will see in section 5, $d_{baseline}$, the measured delay of a certain flow without interference from other flows, demonstrates certain distribution patterns rather than a single delay point, due to the variation of the delay factors. In figure 4, we illustrate a hypothetical probability density function, *f(x)*, of the baseline distribution, $d_{baseline}$. The partial probability density function of the no-queuing case $f(x) * (1 - p)$ is illustrated in figure 5. The partial probability density function of the queuing case, $f(x - d_C) * p$ is equivalent to shifting *f(x)* by d_C and multiplying by *p* as shown in figure 6. As illustrated in figure 7, the estimated delay distribution density function $f^f(x)$ in the presence of the competing flow is derived by adding up partial delay distributions.

$$f^f(x) = f(x - d_C) * p + f(x) * (1 - p) \tag{6}$$

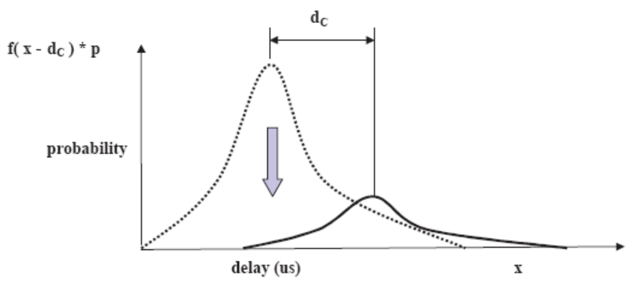

FIGURE 6: Partial Probability Density Function for Queuing Case.

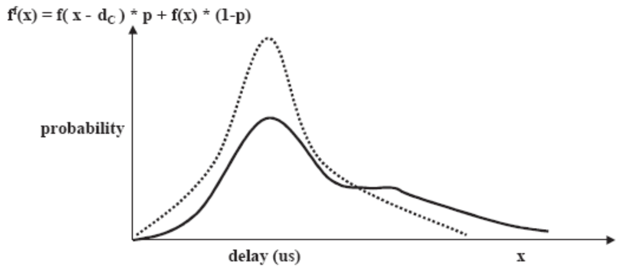

FIGURE 7: Probability Density Function for Output Delay.

In short, if we know the baseline distribution of a new flow and the characteristics (message size and period) of the existing competing flows in the switch, we can estimate the output delay distribution. Finally, the calculated output delay distribution is converted to the cumulative density function $F^f(x)$ for the admission decision. A flow can be admitted, if $F^f(D)$ is greater than DP. For example, in figure 8, the flow S_1 (L_1, P_1, $D_{1,}$ 0.9) is rejected for the delay requirement D_1 but the flow S_1 (L_1, P_1, $D_{1,}$ 0.8) is admitted.

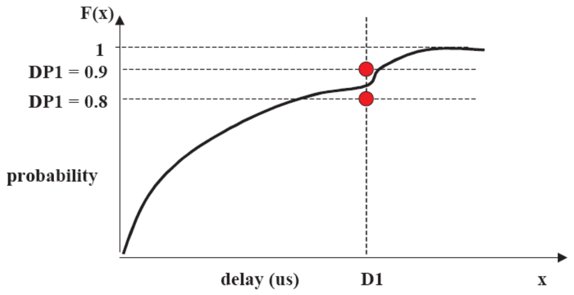

FIGURE 8: Cumulative Density Function for Admission Control.

3.3 The General Case: Multiple Competing Flows

In this section, we consider the generalized case where there are more than two competing flows at a switch. We first introduce a general method called the Benesˇ approach to express the queueing delay of a *G/G/1* queue. We then derive the formula for system multiplexing periodic flows with variable message sizes and variable periods using the Benesˇ approach.

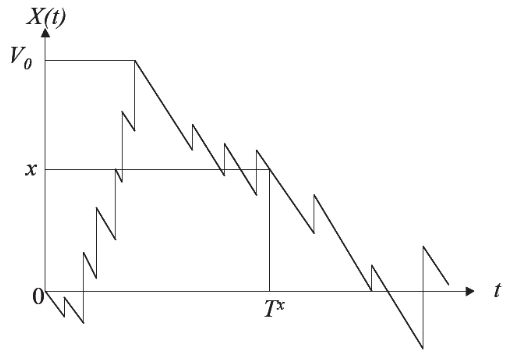

FIGURE 9: A realization of the process *X(t)* with the last exit time T^x.

1) Virtual Waiting Time: The virtual waiting time, also referred to as queueing delay in a system can be obtained by a theorem due to Benesˇ for *G/G/1* queue [4] that has general independent inter-arrival times and general service times. We review the approach here only to the extent that it is essential for our work. Consider a system where a constant rate server with unlimited buffer capacity is submitted work according to a random process. The server capacity is assumed to be 1 unit of work per unit of time and the system is assumed stationary, so that 0 represents an arbitrary time instant. Let $A(t)$, $t \geq 0$, denote the amount of work arriving to the system in the interval [–t, 0), and let V_t be the amount of work still in the system at time *–t*. Define *X(t) = A(t) – t* to be the excess work arriving [–t, 0). Then V_t is given by Reich's formula

$$V_t = \sup_{w \geq t}\{X(w) - X(t)\} \tag{7}$$

In particular,

$$V_0 = \sup_{t \geq 0}\{X(t)\} \tag{8}$$

Let T^x denote the largest value of t such that *X(t) = x* (see figure 9), then the following equivalence can be deducted:

$$\{V_0 \geq x\} \quad \Leftrightarrow \quad \{\exists\ unique\ T^x\ such\ that\ X(T^x) = x$$

$$and\ X(w) < x\ for\ w > T^x\} \tag{9}$$

The complementary distribution function of V_0 can therefore be expressed by the generic Benesˇ principle

$$Pr\{V_0 > x\} = Pr\{T^x \in [0, \infty)\} \tag{10}$$

Employing the definition of T^x from (9) we have

$$\begin{aligned} Pr\{V_0 > x\} &= \int_{u=0}^{\infty} Pr\{u \leq T^x < u + du\} \\ &= \int_{u=0}^{\infty} Pr\{X(u + du) < x \leq X(u) \qquad (11) \\ &\qquad and\ X(w) < x\ for\ w > u\} \qquad (12) \end{aligned}$$

Now, applying relation (7) at the point *t = u* yields

$$\{V_0 = 0\} = \{X(w) \leq X(u)\ for\ w > u\} \tag{13}$$

Applying this to (11), it leads to the Benesˇ formula:

$$Pr\{V_0 > x\} = \int_{u>0} Pr\{X(u) \geq x > X(u + du)\ and\ V_u = 0\} \tag{14}$$

2) *Multiplexing periodic flows with variable message sizes and variable periods*: We address the problem of periodic flows with variable message sizes and variable periods. We refer to this system as $\sum D_i / \sum D_i / 1$ queue. We derive bounds for the queue length distribution of $\sum D_i / \sum D_i / 1$ queue by Benesˇ approach. Suppose *M* flows are sharing a switch port. Then there are 2^M numbers of flow combinations. We denote the set of all flows as C_2^M and a combination of the flows that is a subset flow of C_2^M, as C_m. In a system fed by periodic flows, work arrives discontinuously. Then the probability in the right-hand side of (14) is concentrated on the values of *u* such that *x + u* is an integer number of packet processing time. Let d_{Cm} denote the sum of the delays caused by the packets of the flow set C_m.

$$d_{C_m} = \sum_{i \in C_m} T_i \tag{15}$$

We can thus replace the integral in (14) by a summation and give the virtual waiting time formula for $\sum D_i / \sum D_i / 1$ queue:

$$\begin{aligned} Pr\{V_0 > x\} &= \sum_{d_{C_m} > 0} Pr\{A(d_{C_m} - x) = d_{C_m}\ and\ V_{d_{C_m} - x} = 0\} \qquad (16) \\ &= \sum_{d_{C_m} > 0} Pr\{A(d_{C_m} - x) = d_{C_m}\} \qquad (17) \\ &\qquad \cdot Pr\{V_{d_{C_m} - x} = 0 | A(d_{C_m} - x) = d_{C_m}\} \end{aligned}$$

The first part of equation (17) is computed as follows:

$$\begin{aligned} Pr_{C_m} &= Pr\{A(d_{C_m} - x) = d_{C_m}\} \\ &= \prod_{i \in C_m} \frac{T_i}{P_i} \times \prod_{i \in (C_{2^M} - C_m)} (1 - \frac{T_i}{P_i}) \end{aligned} \tag{18}$$

Pr_{Cm} is the queueing probability where a combination of packets from flow set C_m are queued among the all active flows in the switch and the packets from the rest of the flows ($C_2{}^M$ – C_m) are not queued. The actual corresponding queueing time is less than or equal to d_{Cm}, since the first packet in the queue may be already being served at the time of its arrival. The order of the packets does not affect its queueing time. The second part of equation (17) is replaced by $(1 - \rho_c)^+$ where ρ_c is the conditional arrival intensity at time $d_{Cm} - x$:

$$\rho_c = \sum_{i \in (C_{2^M} - C_m)} \frac{T_i}{P_i(1 - \frac{T_i}{P_i})} = \sum_{i \in (C_{2^M} - C_m)} \frac{T_i}{P_i - T_i} \tag{19}$$

In practice, this quantity is a very close one, since for a high capacity link that multiplexes a large number of flows, the system behaves like a multi-server system for which the empty queue probability is very much closer to 1 than $1 - \rho$. Omission of the condition $\{V_0 = 0\}$ in (14) yields an upper bound for $Pr\{ V_0 > x\}$. The approximation has been shown to be reasonably accurate [22]. This approximation is assumed at the two competing flow cases studied in the earlier section. The notations are summarized in Table 2.

Notation	Explanation
M	the number of existing flows sharing the switch port
C_{2^M}	a set of all M flows
C_m	a subset flow of C_{2^M}
T_i	packet transmission time of flow i (i.e., $L_i/Switch\_Capacity$)
Pr_{C_m}	probability of queueing by a flow set C_m
d_{C_m}	delay caused by C_m

TABLE 2: Notations for Analysis.

The estimated distribution gives an approximation of a worst case delay distribution, since it counts the whole packet processing time of all the packets in the queue, even though the first packet in the queue is already being served. Queueing system studies on multiplexing periodic

flows have been limited to flows with the same packet size [14] ($N * D/D/1$, $\sum D_i /D/1$ queue) in

the past. Equation (17) together with Equations (18) and (19) provides a new method for estimating the queueing delay probability for periodic flows with variable message sizes and variable periods, and for obtaining a probabilistic delay bound.

3.4 The Admission Control Algorithm

The admission control algorithm running on hosts makes a decision if QoS requirements of flows on the host would be violated or not. In this section, we present the admission control algorithm using the queueing analysis described in the previous section. Since measured delay $d_{baseline}$ shows a distribution rather than a point, we cannot directly apply the complementary distribution $Pr\{V_0 > x\}$ to the admissibility test. Instead, the partial probability density function $f'(x)$ is obtained in equation (20), using its queueing probability Pr_{Cm} and the corresponding delay d_{Cm} for each queueing flow set C_m,.

$$f'(x) = Pr_{Cm}\, f(x - d_{Cm}) \tag{20}$$

The partial probability density functions are summed up to be the output delay distribution $f^f(x)$. The output delay distribution is converted to the cumulative density function $F^f(D)$ to see if the deadline is satisfied with the required probability. The probability that the delay is less than the deadline D is evaluated from the final cumulative density function F^f as below (equation 22):

$$F^f(x) = \int_0^x \sum_{m \in C_{2M}} Pr_{C_m} f(t - d_{C_m}) dt \tag{21}$$

$$F^f(D) > DP \tag{22}$$

The algorithm is particularly complex to evaluate, since the number of possible queueing combinations grows exponentially with the number of competing flows. For each possible competing flow combination, the algorithm needs computation process and memory usage to modify the partial probability density function (to be shifted and to be multiplied) and to convert it to the cumulative density function.

Algorithm 1 Admissibility Test Algorithm

```
$F^f(x) \leftarrow$ baseline distribution of flow $s$
$F^f_{req} \leftarrow 0$ // initial probability for admission decision
foreach $m \in C_{2M}$ // for each subset flows
        // Calculate the probability by a set $C_m$
        $Pr_{C_m} = \prod_{i \in C_m} \frac{T_i}{P_i} \times \prod_{i \in (C_{2M} - C_m)} (1 - \frac{T_i}{P_i})$
        if ($Pr_{C_m} < \epsilon$)
            break // Speed up the decision
        $d_{C_m} = \sum_{i \in C_m} T_i$ // Calculate the caused by a set $C_m$
        $F^f_{req} += Pr_{C_m} \times F(D - d_{C_m})$
        if ($F^f_{req} > DP$)
            goto Decision //condision satisfied already
Decision:
        if (no reject signal received and $F^f_{req} > DP$)
            s is admitted
        else
            if $s$ is on this host
                $s$ is rejected
            else
                send reject control message
```

To make this approach practical, we made a couple of improvements to the algorithm. First, we observed that the packet queuing probability, due to a large number of other competing flows is very small when the stable network condition is satisfied. In most of the experiment settings, the packet queuing probability due to more than five competing flows was less than 10^{-10}. Therefore, the computation of those combinations can be waived. More importantly, we found a derivation that enables us to make the admission decision without handling partial probability density function modifications and the cumulative density function conversion. In the following derivation (equation (23)), the admission decision is made by using the cumulative density function of the baseline distribution. Only one point ($D - d_C$) of the baseline distribution needs to be evaluated for each modification of the competing flow combinations.

$$\begin{aligned} F^f(D) &= \int_0^D \sum_{m \in C_{2M}} Pr_{C_m} f(t - d_{C_m}) dt \\ &= \sum_{m \in C_{2M}} Pr_{C_m} \int_0^D f(t - d_{C_m}) dt \\ &= \sum_{m \in C_{2M}} Pr_{C_m} F(D - d_{C_m}) \qquad (23) \end{aligned}$$

For a requested flow to be admitted, the same admission decision process should also be made by all hosts that have existing flows competing for the same switch output port.

4. SOFTWARE DESIGN AND IMPLEMENTATION

We have implemented the probabilistic admission control software on the Windows operating system. As shown in figure 10, the proposed admission control software is implemented in the application layer and the packet classifier and the token bucket regulation-based packet controller in the Windows operating system kernel are used to maintain the flow period.

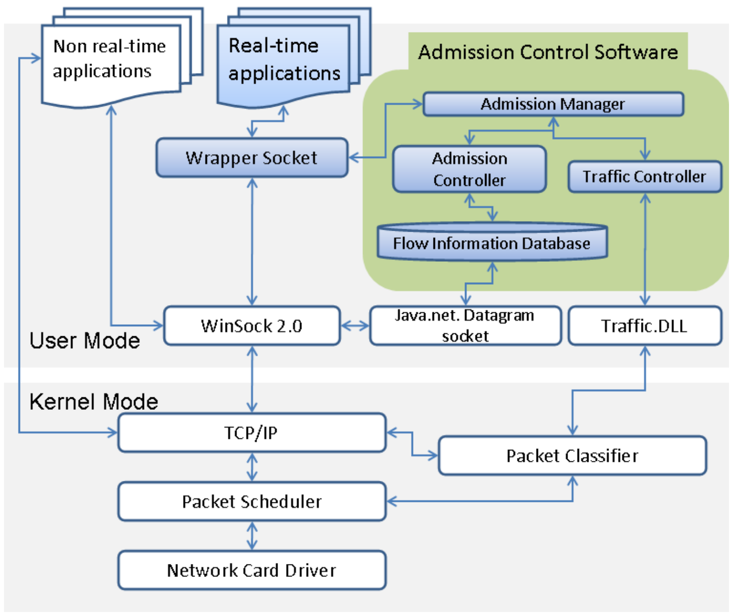

FIGURE 10: Windows Based soft real-time system Architecture.

The admission control software comprises the admission manager, the admission controller, the traffic controller, and the flow information database. The admission manager provides a registration and deregistration method to the traffic controller as well as an interface to the admission controller. If a new flow request is admitted, the admission manager registers the flow to the traffic controller. The traffic controller is responsible to relay flows to the packet scheduler in the kernel through the traffic.DLL calls. The flow informs the packet control method to be used for the packet scheduler. It also creates a filter as specified by the specification to instruct the packet classifier to filter the list of packets. The flows are controlled according to the flow specification described by the admission manager. The admission controller performs admissibility test for a new flow request based upon the flow specification and the existing flow information from the flow information database. If the flow request is admissible, the admission controller creates a flow and adds the new flow to the flow information database. The flow information database maintains a consistent image of the network topology and the existing flow information of the entire network control system. It also interfaces to the broadcast socket to send a broadcast message of flow addition to other control hosts in order to update their flow information database. The admission control software is used by the real-time applications via Wrapper Socket API calls that are the Java-based wrapper APIs implemented above Winsock 2. The Wrapper Sockets pass the flow specification to the admission manager along with the source and destination IP addresses, destination port, and QoS enable indication. It relays the application flows to the Winsock 2 according to the admission decision.

5. EXPERIM ENTS AND RESULTS

To validate the efficiency of the proposed probabilistic admission control approach, we have conducted extensive experiments on the real networks. In this section, we present the experimental settings design and results. As illustrated in figure 11, 9 PCs are connected to a 12 port Intel express 520T fast Ethernet switch. A test flow generator generates a test message flow to a test flow receiver. Other 7 PCs are used to inject competing traffic flows. Competing flows are generated by seven other hosts and ten flows are generated per host. Competing flows are sent to the same receiving host as the test flow, ensuring that all competing flows use the same output port of a switch. A combined time delay of the hosts, network, and switch buffer is measured using time-stamps derived from the TrueTime^{TM} [1] clock synchronization tool that provides one microsecond time resolution. To capture variability of flows both in terms of message size and period, we utilize an extensive set of experimental parameters. The experimental parameters, such as system environment and flow specifications, are summarized in Table 3. Although the algorithm can be applied to the flows with various periods with different message sizes, we have conducted the experiments with the same message size and period to illustrate the behavior of the delay as a function of message size and period. For the experimental parameters, we choose the message size less than the maximum packet fragmentation size (1500 bytes) in order not to introduce additional unnecessary complexity. We also keep the period greater than 10 milliseconds due to host delay stability issues (the flow control resolution of Windows). In a practical control system, most control message sizes are less than the maximum packet fragmentation size and the periodicity constraints are greater than tens of milliseconds [2].

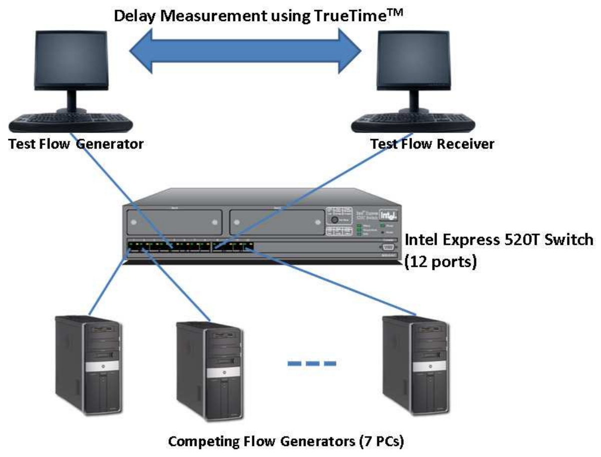
Delay Measurement using TrueTime™
Test Flow Generator
Test Flow Receiver
Intel Express 520T Switch (12 ports)
Competing Flow Generators (7 PCs)

FIGURE 11: Testbed Configuration.

Parameters	Values
System Environment	
Number of Switches between hosts Switch Capacity (Mbps)	1 100
Baseline Flow	
Message size (bytes) Message period (ms)	50, 500, 1400 50, 200, 500
Competing Flows	
Number of competing hosts Number of flows per competing host Message size (bytes) Message period (ms)	7 10 50, 500, 1400 50, 200, 500
Test Flow QoS Required	
Acceptable Delay (μs) Min Prob Actual Delay $<$ Acceptable Delay	700, 800, 900 0.90 $\sim$ 0.99
Windows Flow Control Mechanism	
Max Packet Size (bytes) Token Rate Token Bucket Size Peak Bandwidth	1526 (for Ethernet) Based on period Based on message size Based on period and message size

TABLE 3: Experiment Parameters.

5.1 Baseline and Competing Flow Experiments

We first generate the baseline delay probability distributions that characterize a single sender/receiver host pair without any other competing flows. This baseline distribution is used to estimate the theoretical distribution with competing flows. We then conduct experiments to measure delay distributions with competing flows to validate the theoretical distribution estimations. Each experiment is performed for 30 minutes with 5 minute warm-up interval. The delay distributions captured on the receiver are presented as PDF (Probability Density Function) and CDF (Cumulative Density Function).

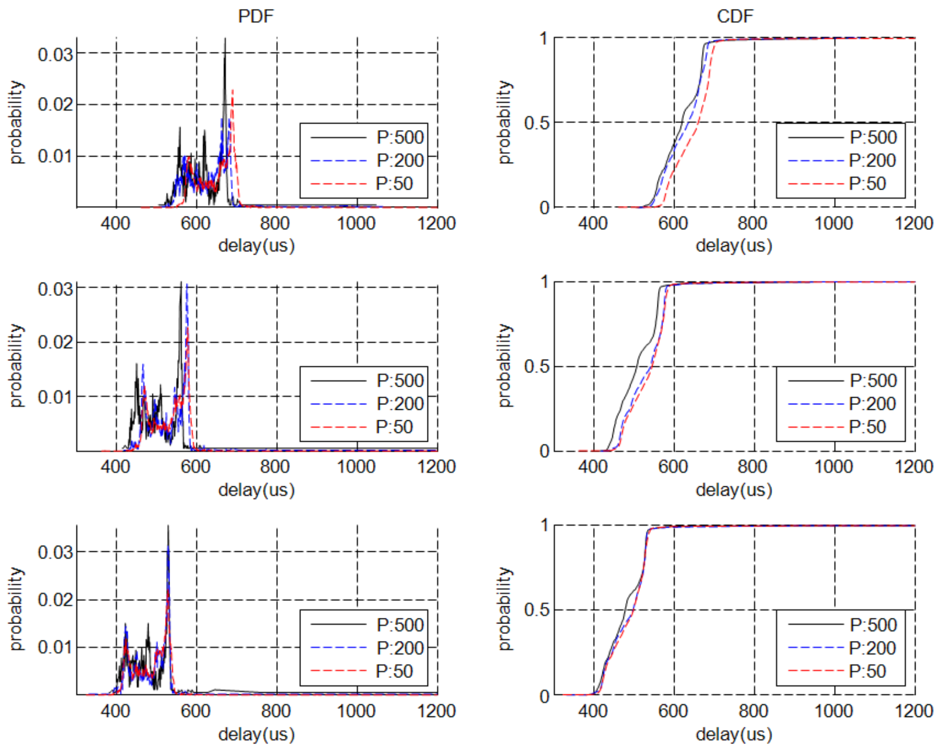

FIGURE 12: Baseline Distributions: PDFs and CDFs: Fixed Message Size (1400/500/50B from the top row), Variable Periods.

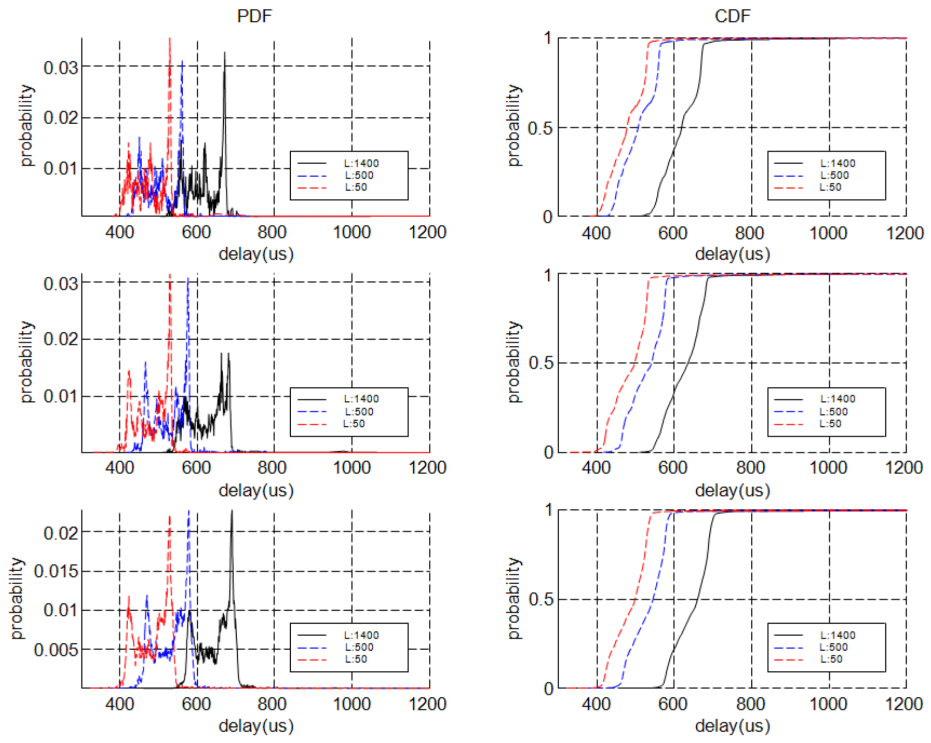

FIGURE 13: Baseline Distributions: PDFs and CDFs: Fixed Period (500/200/50 ms from the top row), Variable Message Sizes.

Figures 12 and 13 present the distribution of baseline measurements. In Figure 12, PDFs and CDFs are plotted for message sizes of 1400, 500, and 50 bytes, respectively from the first row. Each plot is compared with variable periods of 500, 200, and 50 ms. The PDF results illustrate that there are multiple modes on the distribution. These distributions look different from any well-known distributions that can be analytically tractable. Therefore, it seems infeasible to model this distribution parametrically. The distributions are, however, quite consistent with the entire experiment and are stable enough to use them as the basis of the estimated distributions with computing flows. Figure 13 shows the PDF and CDF results of various packet sizes (1400, 500, and 50 bytes) with a fixed message period per plot. Each plot illustrates message periods of 500, 200, and 50 ms. The CDF results show that the larger message size has more processing delay on the host and switch due to the longer transmission time that causes larger application-to-application delays. An observation from the experiment confirms that the CDF distribution shapes are consistent with the same period across the message sizes. Hence, if we have one baseline distribution for a flow, we can use the distribution for the same message size with different periods and we may further estimate the baseline distribution of the different message sizes without measurements. Using the same parameters as the baseline configuration for the test flow, we ran simultaneous competing flows to obtain a delay distribution that accounts for message queueing in the switch. The measured distribution is used to compare with the estimated distribution to assess its accuracy.

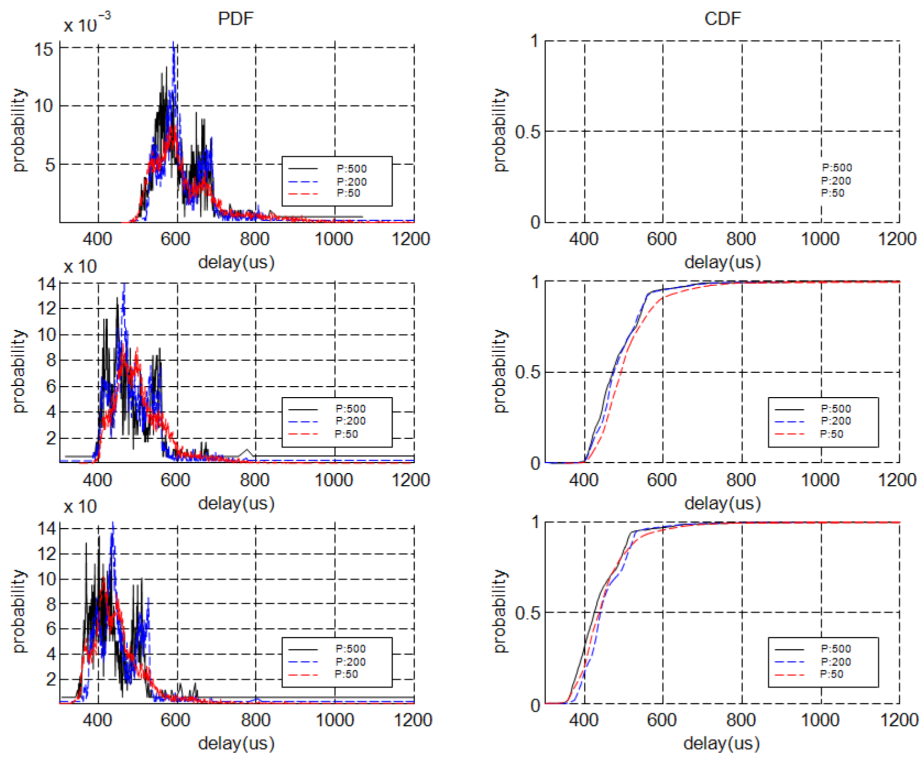

FIGURE 14: Competing Flows: PDFs and CDFs-Fixed Message Size (1400/800/50B from the top row), Variable Period.

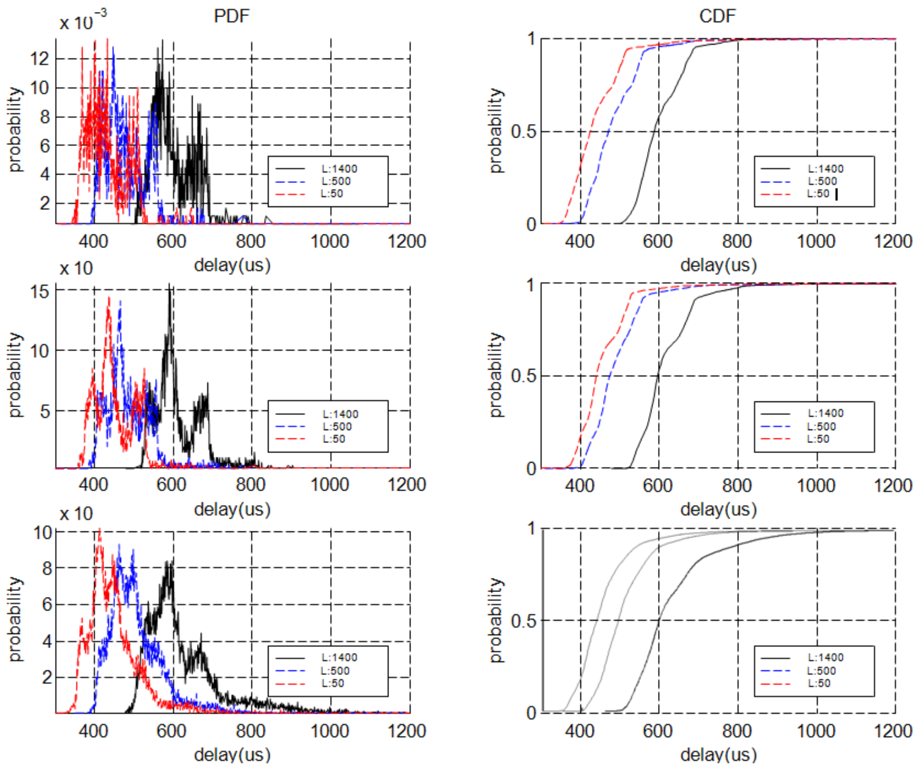

FIGURE 15: Competing Flows: PDFs and CDFs-Fixed Period (500/200/50ms from the top row), Variable Message Size.

The delay distributions with competing flows are shown in figures 14 and 15. In figure 14, PDFs and CDFs are plotted for message sizes of 1400, 500, and 50 bytes, respectively, from the first row. Each plot is compared for variable periods of 500, 200, and 50 ms. figure 15 presents the results of various packet sizes (1400, 500, and 50 bytes) with a fixed message period per plot. Each plot illustrates with message periods of 500, 200, and 50 ms. The experiment uses 7 competing flow generators and each host has 10 competing flows that makes 71 flows in total including the test flow. To ensure the flow independence and to reduce the effect of phase synchronization on the periodic flows, competing flows are generated with different start times that are made to be much bigger than the maximum period (if the start time difference is less than the maximum period, it may end up phase synchronization with the repeating experiments). To have solid statistical results, the experiments are performed repeatedly (more than 200 times). The experiments with competing flows clearly show that flows with longer messages and shorter periods experience longer delays with higher probability due to queueing events in the switch. In figure 14, shorter periods tend to result in a high probability of extreme delay for the fixed message sizes. i.e., PDFs are more widely distributed (compared to baselines) and in CDFs, it reaches to one slowly especially in high probability regions. In figure 15, the distributions of larger messages sit on the right-hand side of the smaller messages and the gaps are bigger than the case for baselines since that includes not only transmission delay of its message but also queueing delays.

5.2 Algorithm Validation

The measurements of various baseline distributions are used to estimate delay distributions that account for queueing in the switch for the same number and parameters of flows as in an earlier section. The estimated distributions are compared with the actual competing flow experiment distributions jointly with baselines in figure 16. The figures are shown for the high probability region (> 0.9) that aligns with the actual real-time requirements. The results show that the estimated distributions with competing flows approach to the experimental distributions especially in the higher probability regions. It is also observed that the estimated distributions are more conservative than the experiment distribution because it considers the worst case distribution. These experimental results in the real implementation validate the proposed approach.

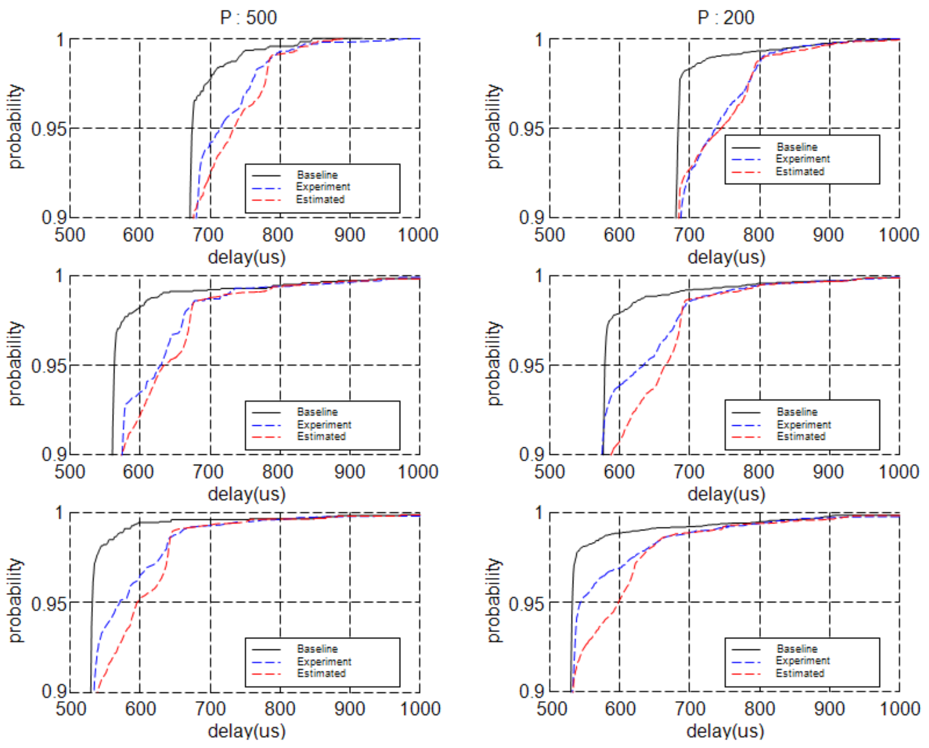

FIGURE 16: Delay CDFs from the Proposed Algorithm with Variable Periods.

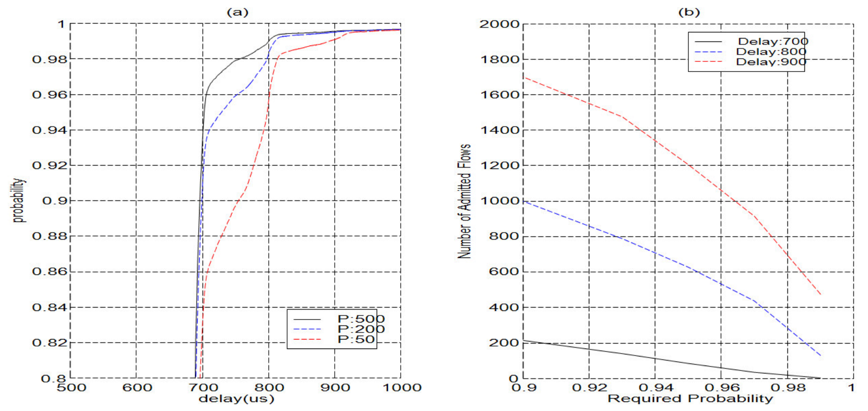

FIGURE 17: Estimated distributions (a) Estimated output distribution for variable period, (b) Number of admitted flows.

The trend of the estimated distributions for different parameters is shown in figure 17 (a). Figure 17 (b) also shows the trend in the number of admitted flows in various QoS requirements by successively admitting flows to a system of existing flows. The experiment was performed with a message size of 1400 bytes and a period of 200 *ms*. Since the algorithm admits flows as long as the bandwidth sum is less than the switch capacity and the baseline gives enough probability for the longer delay requirements, we test for shorter delay requirements (700, 800, 900 µs) to show the accuracy. The result shows that the flow admissibility increases for the less requested probability or the longer delay requirements. This indicates that the utilization can be increased by relaxing QoS requirements. The utilization gain is linear rather than exponential since the flows are periodic.

6. CONSLUSIONS

We have presented a novel and efficient probabilistic admission control approach to support soft real-time control applications over switched Ethernet. Our approach enables real-time application-to-application QoS management over switched Ethernet without sophisticated packet scheduling or resource reservation mechanisms in Ethernet switches or middleware on end hosts. Application-to-application delay is estimated based on the delay distribution of baseline measurements and queueing analysis with competing flow information. As part of our contributions, we have provided a new and efficient method to identify and estimate the queueing delay probability inside Ethernet switches for heterogeneous periodic control system applications with variable message size and period. This queueing analysis is interesting in itself. We have implemented the probabilistic admission control algorithm on the Windows operating system, and validated its efficiency through extensive experiments.

7. REFERENCES

[1] TrueTime, Inc. http://www.truetime.com/.
[2] Worldwide Plant Automation Systems Outlook-Market Analysis and Forecast Through 2010. Automation Research Corporation, 2005.
[3] A. K. Atlas and A. Bestavros. Statistical Rate Monotonic Scheduling. In 19th IEEE Real-Time Systems Symposium, Madrid, Spain, December 1998.
[4] V. E. Benes˘. General Stochastic Processes in the Theory of Queues. Addison-Wesley, 1963.
[5] R. Caponetto, L. L. Bello, and O. Mirabella. Fuzzy Traffic Smoothing: Another Step towards Real-Time Communication over Ethernet Networks. In 1st RTLIA, Vienna, Austria, 2002. IEEE Computer Society.
[6] T. Chiueh and C. Venkatramani. Supporting real-time traffic on Ethernet. In Proceedings of IEEE real-time Systems Symposium,1994.
[7] R. Court. Real-Time Ethernet. ACM Computer Communications, 15(3):198–201, Apr. 1992.

[8] Xing Fan and Magnus Jonsson. Guaranteed Real-Time Services over Standard Switched Ethernet. In Proceedings of the IEEE Conference on Local Computer Networks 30th Anniversary (LCN,05), Sydney, Australia, 2005. IEEE Computer Society.
[9] Robert Janowski, Piotr Krawiec, and Wojciech Burakowski. On assuring QoS in Ethernet access network. In Proceedings of the Third International Conference on Networking and Services ICNS'07, Athens, Greece, 2007. IEEE Computer Society.
[10] S.-K. Kweon and K. Shin. Real-Time Communication over Ethernet with Adaptive Traffic Smoothing. In RTAS '00: Proceedings of the Sixth IEEE Real-Time Technology and Applications Symposium, Washington, DC, USA, 2000. IEEE Computer Society.
[11] S. Lima, P. Carvalho, and V. Freitas. Distributed Admission Control in Multiservice IP Networks: Concurrency issues. Journal of Communications, 1(3):1–9, June 2006.
[12] J. Loeser and H. Haertig. Low-latency Hard Real-Time Communication over Switched Ethernet. In Proceedings of the 16th Euromicro Conference on Real-Time Systems (ECRTS 04), Catania, Italy, 2004. IEEE Computer Society.
[13] D. Loy and R. Schmalek. Thoughts About Redundancy in Fieldbus Systems Anchored in OSI Layer-4 and Applied to the Lontalk Protocol on Neuron-Based Network Nodes. In IEEE International Workshop on Factory Communication Systems, October 1995.
[14] I. Norros, J. W. Roberts, A. Simonian, and J. T. Virtamo:. The superposition of variable bit rate sources in an ATM multipler. IEEE Journal of Selected Areas on Communication, 9(3):378–387, 1991.
[15] P. Pedreiras, L. Almeida, and P. Gai. The ftt-ethernet protocol: Merging flexibility, timeliness and efficiency. In Euromicro ECRTS'02, Vienna, Austria, 2002. IEEE Computer Society.
[16] P. Pedreiras, R. Leite, and L. Almeida. Characterizing the Real-Time Behavior of Prioritized Switched-Ethernet. In 2nd RTLIA, Porto, Portugal, 2003. IEEE Computer Society.
[17] D. W. Pritty, J. R. Malone, S. K. Banerjee, and N. L. Lawrie. A real-time upgrade for Ethernet based factory networking. In Annual Conference of the IEEE Industrial Electronics Society (IECON), 1995.
[18] A. Raha, S. Kamat, and W. Zhao. Guaranteeing End-to-End Deadlines in ATM networks. In 15th International Conference on Distributed Computing Systems, 1995.
[19] A. Raha, S. Kamat, and W. Zhao. Admission Control for Hard Real-Time Connections in ATM LANs. In IEEE INFOCOM, San Francisco, CA, March 1996.
[20] Automation Strategies Report. Ethernet-Based Control Network Strategies. Automation Research Corporation, Oct. 1997.
[21] Y. Shimokawa and Y. Shiobara. Real-Time Ethernet for industrial applications. In Annual Conference of the IEEE Industrial Electronics Society (IECON), 1985.
[22] V. Sivaraman and F. Chiussi. Providing End-to-End Statistical Delay Guarantees with Earliest Deadline First Scheduling and Per-Hop Traffic Shaping. In IEEE INFOCOM, 2000.
[23] T. S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L. C. Wu, and J. W. S. Liu. Probabilistic performance guarantee for real-time tasks with varying computation times. In IEEE Real-Time Technology and Applications Symposium, Chicago, IL, May 1995.
[24] C. Venkatramani and T. Chiueh. Design, Implementation, and Evaluation of a Software-based Real-Time Ethernet Protocol. In ACM SIGCOMM, Cambridge, MA, August 1995.

Achieving Energy Proportionality In Server Clusters

Xinying Zheng zxying@mtu.edu
Ph.D Candidate /Electrical and Computer Engineering
Michigan Technological University
Houghton, 49931, US

Yu Cai cai@mtu.edu
Associate Professor /School of Technology
Michigan Technological University
Houghton, 49931, US

Abstract

Green computing is a hot issue that has received a great amount of interests in the past few years. Energy proportionality is a principal to ensure that energy consumption is proportional to the system workload. Energy proportional design can effectively improve energy efficiency of computing systems. In this paper, an energy proportional model is proposed based on queuing theory and service differentiation in server clusters, which can provide controllable and predictable quantitative control over power consumption with theoretically guaranteed service performance. Further study for the transition overhead is carried out corresponding strategy is proposed to compensate the performance degradation caused by transition overhead. The model is evaluated via extensive simulations and justified by the real workload data trace. The results show that our model can achieve satisfied service performance while still preserving energy efficiency in the system.

Keywords: green computing, energy proportional, performance differentiation, transition overhead

1. INTRODUCTION

Green computing is to support personal and business computing needs in a green and sustainable manner, such as minimizing strain and impact on resources and environment. Computing systems, particularly enterprise data centers and high-performance cluster systems consume a significant amount of energy, thus placing an increasing burden on power supply and operational cost. For example, the power consumption of enterprise data centers in the U.S. doubled between 2000 and 2005, and will likely triple again in a few years [1]. In 2005, US data centers consumed 45 billion kWH, which was roughly 1.2 percent of the total amount of US electricity consumption, resulting in utility bills of $2.7 billion [2]. In 2006, the U.S. Congress passed bills to raise the IT industry's role in energy and environmental policy to the national level [3]. Furthermore, it is estimated that servers consume 0.5 percent of the world's total electricity [4], which if current demand continues, is projected to quadruple by 2010. Some analysts predicted that IT infrastructure power usage will soon cost more than the hardware itself [5].

Many of the existing works on power management in server clusters rely heavily on heuristics or feedback control [6][7][8][9]. An important principle in green computing is to ensure energy consumption proportionality, which states that the energy consumption should be proportional to the system workload [10]. For example, when there is no or little workload, the system should consume no or little energy; when workload increases, energy consumption should increase proportionally, until the system reaches the full workload. This idea can effectively improve the energy efficiency in real-life usage. Energy proportionality is relatively hard to be achieved on a standalone server because of hardware constraints. However, it is possible to achieve energy proportionality on a server cluster, since we can control the number of active and inactive nodes in a server cluster.

In this paper, we propose an energy proportional model in a server cluster and study its performance in both single and multiple classes' scenarios. We further investigate the transition overhead based on this model. The simulation results show that the energy proportional model can provide controllable and predictable quantitative control over power consumption with theoretically guaranteed service performance.

The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 introduces the energy proportional model. Performance metrics and servers allocation strategy are introduced in section 4. Section 5 evaluates the model and discusses the transition overhead influence, a strategy is also proposed to compensate the transition overhead in this section, the model is further evaluated based on the real workload data trace, and the last section concludes the paper.

2. RELATED WORK

In literatures, green computing is often related to terms like green IT, sustainable computing, energy efficiency, energy saving, power aware, power saving, and energy proportional. In this section, we review relevant techniques commonly used on single server and server clusters.

A. Single Server

The green computing techniques for a single server focus on microprocessors, memories and disks. Current microprocessors allow power management by dynamic voltage and frequency scaling (DV/FS). DV/FS works because reducing the voltage and frequency provides substantial savings in power at the cost of slower program execution. Some researches tie the scheduler directly to DV/FS [11][12][13]. Most works deal exclusively with meeting real-time scheduling deadlines while conserving energy.

Traditionally, many power management solutions rely heavily on heuristics. Recently, feedback control theoretical approaches for energy efficiency have been proposed by a number of researchers. On a single server, recent works [14][15] proposed power control schemes based on feedback control theory. Femal et al. [16] developed an algorithm based on linear programming. In [8], a control theoretical power management scheme on standalone servers was proposed. The feedback control theory is better than the traditional techniques by providing high accuracy and stability.

Thermal management is another issue in power-aware computing, since temperature is a by-product of power dissipation [17]. Recent research demonstrated that dynamic thermal management (DTM) can respond to thermal conditions by adaptively adjusting a chip power consumption profile on the according to feedback from temperature sensors [14] [18].

Research work on memory is often combined with processors and disks. In [19], the authors used open-loop control to shift power between processor and memory to maintain a server power budget. In [20], they proposed a solution to store pages and reliability data in idle RAM instead of using slow disk. A large portion of the power budget of servers goes into the I/O subsystem, the disk array in particular. Many disk systems offer multiple power modes and can be switched to a low power mode when not in use to achieve energy saving. Such techniques had been proposed in [21][22]. Sudhanva et al. [23] presented a new approach called DRPM to modulate disk speed dynamically, and a practical implementation was provided for this mechanism.

B. Server Clusters

In recent years, power management has become one of the most important concerns on server clusters. Some methods proposed on a single server can be extended to server clusters. In [24][25], the authors presented similar ways of applying DV/FS and cluster reconfiguration, using threshold values, based on the utilization of the system load to keep the processor frequencies as low as possible, with less active nodes. In [9], the authors extended the feedback control scheme to clusters. Power has been used as a tool for application-level performance requirements. Sharma et al. [26] proposed feedback control schemes to control application-level quality of service requirements. Chen et al. [27] presented a feedback controller to manage the response time in server clusters. Some researchers applied DTM on an entire data center rather than individual servers or chips. In [28], the authors laid out policies for workload placement to promote uniform temperature distribution using active thermal zones.

Vary-On Vary-off (VOVF) is a dynamic structure configuration mechanism to ensure energy-aware computing in server clusters, which turns nodes on and off to adjust the number of active servers by the workload. Other work had been carried out based on VOVF [29][30][28]. In [31], The authors proposed a method to reduce network energy consumption via sleeping and rate adaptation by combining VOVF and DV/FS. Another group developed power saving techniques for connection oriented servers [32]. The authors tested server provisioning and load dispatching on the MSN instant messaging framework, and evaluated various load skewing techniques to trade off energy saving and quality of service.

Virtualization is another key strategy to reduce power consumption in enterprise networks. With virtualization, multiple virtual servers can be hosted on less but more powerful physical servers, using less electricity [33]. In [34], researchers developed methods to efficiently manage the aggregate platform resources according to the guest virtual machines (VM) of relative importance (Class-of-Service), using both the black-box and the VM-specific approach. Hu et al. [35] used live migration of virtual machines to transfer load among the nodes on a multilayer ring-based overlay. In [4], researchers scheduled virtual machines in a computer cluster to reduce power consumption via the technique of Dynamic Voltage Frequency Scaling (DVFS). An economy driven energy and resource management framework was presented for clusters in [36]. Each service ”bids” for resources as a function of delivered performance. In [37], researchers formulated the problem as a cooperative game, and used game theory to find the bargaining point.

The energy-related budget has accounted for a large portion of total storage system cost of ownership. Some studies tried multispeed disks for servers [23][38]. Other techniques were introduced to regulate data movement. For example, the mostly used data can be transferred to specific disks or memory, thus other disks can be set to a low power mode [39].

3. ENERGY PROPORTIONAL MODEL

A. Energy Proportional Model on a Single Server

The energy proportional model states that energy consumption *P* should be proportional to the workload λ , while ensuring service performance.

$$P = a * \lambda + b \tag{1}$$

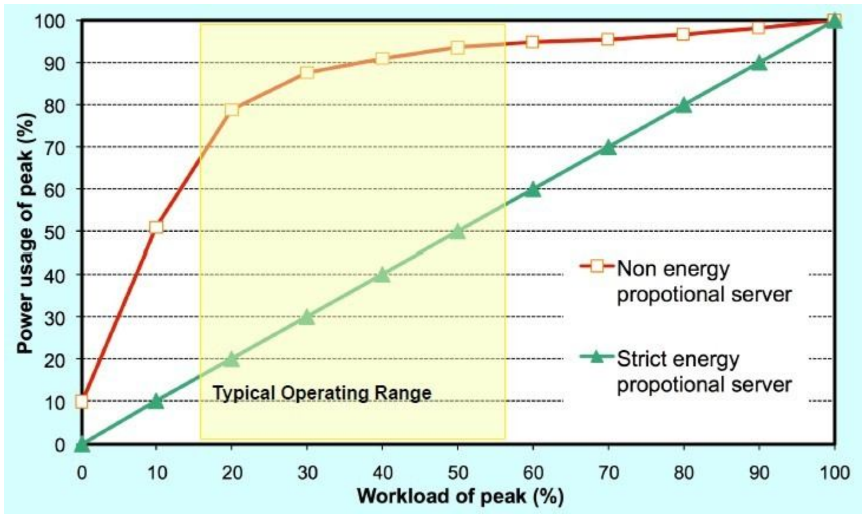

Fig. 1. The energy consumption curves of non-energy proportional server and strict energy proportional server.

Figure.1 conceptually illustrates the energy consumption curve in non-energy proportional servers and energy proportional servers. The typical server operating range is between 10% - 60%. We can see that in a non-energy proportional server, it still consumes about half of its full power when doing virtually no work [10]. Energy proportional server ideally consumes no power when idle (*b* = 0), nearly no power when very little work is performed, and gradually more power as the activity level increases. Energy-proportional designs would enable large energy savings on servers. However, most servers nowadays are CPU, memory and hard disk intensive servers. The energy consumption of CPU is almost linear to its utilization [32]. But memory and hard disks are nonlinear energy consumption components. As a result, energy proportionality is not easy to be achieved on a standalone sever because of the hardware constraints.

B. Energy Proportional Model on Server Clusters

It is more feasible to achieve energy proportionality in a server cluster. Most computing systems nowadays have at least two modes of operation: an active mode when the system is working and an idle mode when the system is inactive and consumes little energy. Some researchers proposed to have finer-grained power modes, running at low speed and with lower power supply voltage. It is known that to quantitatively control energy consumption, one feasible way is to adaptively and dynamically control the number of servers running in active and inactive modes according to system workload.

For simplicity, we assume all the servers in the cluster are identical nodes. On typical web servers and web clusters, system workload can be described by the request arrival rate λ . Let *M* be the total number of servers in the cluster, and Λ be the maximum arrival rate for the cluster. $\sum m$ is the total number of active servers. The total energy consumption of a server cluster is:

$$P = \sum m * P_{ac} + \left(M - \sum m\right) * P_{in} \tag{2}$$

P_{ac} is the power consumption of fully active nodes; P_{in} is the power consumption of inactive nodes. Based on the energy proportional model, we have:

$$\frac{P}{\lambda} = \frac{P_{\max}}{\Lambda} * r \tag{3}$$

where $P_{\max} = M * P_{ac}$. r is a parameter, which adjusts the energy consumption curve in Figure 1. The rationale of using parameter *r* is as follows. Ideally the *r* is set to *r* =1 where energy consumption is strictly proportional to workload. However, we can adjust it to satisfy different performance constraints. With the help of (2), we can rewrite equation (3) as:

$$\sum m = (\frac{P_{ac}}{\Lambda / M} * r - M * P_{in}) / (P_{ac} - P_{in}) \tag{4}$$

Here Λ/M is the maximum jobs that a single cluster node can handle. Ideally P_{in} = 0, which indicates that a server consumes no energy when it is running on an inactive mode. For simplicity, we suppose P_{in} = 0 in this paper, this assumption will not affect the performance of our model. We finally achieve that the total number of active servers $\sum m$ is determined by the system workload λ :

$$\sum m = \frac{\lambda}{\Lambda/M} * r \tag{5}$$

The number of servers may not be an integer based on (5). We will set the integer no less than $\sum m$, which is the minimal number of servers to run in fully active mode.

4. SERVERS ALLOCATION BASED ON ENERGYPROPORTIONAL MODEL

An important task of energy aware computing is to achieve energy efficiency while ensuring performance. In this section, we will describe how to allocate servers according to workload, while ensuring quality of services (QoS) metrics.

A. Performance Metrics

One important and commonly used QoS metric on Internet servers is slowdown, which is defined as the division of waiting time by service time. Another commonly used performance metric is request time which is the sum of waiting time and service time. We choose slowdown and request time as performance metrics in our model because they are related to both waiting time and service time.

Our theoretical framework is built along the line of the previous service differentiation models presented in [40][41][42][43]. In our network model, a heavy-tailed distribution of packet size is used to describe web traffic. Here we assume that the service time is proportional to the packet size.

The packet inter-arrival time follows exponential distributed with a mean of *1/λ*, where λ is the arrival rate of incoming packets. A set of tasks with size following a heavy-tailed Bounded Pareto distribution are characterized by three parameters: α the shape parameter; *k*, the shortest possible job; *p*, the upper bound of jobs. The probability density function can be defined as:

$$f(x) = \frac{1}{1-(k/p)^{\alpha}} \alpha k^{\alpha} x^{-\alpha-1} \tag{6}$$

where, α , $k > 0$, $k \le x \le p$. If we define a function:

$$K(\alpha, k, p) = \frac{\alpha k^{\alpha}}{1-(k/p)^{\alpha}} \tag{7}$$

then we have:

$$E[X]=\int_k^p f(x)dx=\begin{cases}\dfrac{K(\alpha,k,p)}{K(\alpha-1,k,p)} & if\,\alpha\neq 1;\\ (\ln p-\ln k)K(\alpha,k,p) & if\,\alpha=1.\end{cases} \tag{8}$$

Similarly, we can derive $E[X^2]$ and $E[X]$

$$E[X^2]=\int_k^p f(x)x^2dx=\frac{K(\alpha,k,p)}{K(\alpha-2,k,p)} \tag{9}$$

$$E[X^{-1}]=\int_k^p f(x)x^{-1}dx=\frac{K(\alpha,k,p)}{K(\alpha+1,k,p)} \tag{10}$$

According to Pollaczek-Khinchin formula, the average waiting time for the incoming packets is:

$$E[W]=\frac{\lambda E[X^2]}{2(1-\lambda E[X])} \tag{11}$$

We can derive a closed-form expression of the expected slowdown in an M/G/1 queue on a single Internet server.

$$E[S]=E[W]E[X^{-1}]=\frac{\lambda E[X^2]E[X^{-1}]}{2(1-\lambda E[X])} \tag{12}$$

The expected request time with the incoming job rate is:

$$E[R]=E[W]+E[X]=\frac{\lambda E[X^2]}{2(1-\lambda E[X])}+E[X] \tag{13}$$

B. Servers Allocation for a Single Class

In this section, we assume all the incoming requests are classified into just one class. We want to ensure the QoS metrics based on different workload λ. For example, the expected request time of the incoming jobs should stay within a threshold, $E[R]<\beta$. We assume $\sum m$ is the number of active server nodes handling the incoming requests. When using a round-robin dispatching policy, the packet arrival rate of a node is $\lambda/\sum m$. The expected request time in a server cluster can be calculated as:

$$E[R]=\frac{\lambda E[X^2]}{2\left(\sum m-\lambda E[X]\right)}+E[X]<\beta \tag{14}$$

Based on the above energy proportionality (5), equation (14) can be re-written as:

$$E[R]=\frac{E[X^2]}{2(r*M/\Lambda-E[X])}+E[X]<\beta \tag{15}$$

It is easy to observe that request time is not depending on workload, we can just adjust parameter *r* to satisfy different performance thresholds.

C. Servers Allocation on Service Differentiation

Now we study server allocation schemes for service differentiation. In a cluster system, the incoming requests are often classified into *N* classes. Each class may have different QoS requirements. We assume m_i is the number of active server nodes in class i, and λ_i is the arrival rate in class i. The expected slowdown of class i in a server cluster can be calculated as:

$$E[S_i] = \frac{\lambda_i E[X^2]E[X^{-1}]}{2\left(m_i - \lambda_i E[X]\right)} \tag{16}$$

Here we choose not to use request time as a performance metric for service differentiation because of its overly complicated mathematical expression. However, each class should satisfy the request time constraint. Obviously the results presented in this paper will not be affected by the selection of performance metrics.

We adopt a relative service differentiation model where the QoS factor of slowdown between different classes are based on their predefined differentiation parameters.

$$\frac{E[S_i]}{E[S_j]} = \frac{\delta_i}{\delta_j} \tag{17}$$

Where $1 \le i, j \le N$:

We assume class 1 is the highest class and set $0 < \delta_1 < \delta_2 < \cdots < \delta_N$, then higher classes receive better service, i.e., lower slowdown [39].

Based on the above energy proportionality and service differentiation model, according to formula (5)(17), we can derive the server allocation scheme in a cluster system as

$$m_i = \lambda_i E[X] + \frac{\tilde{\lambda}_i \sum_{i=1}^{N} \lambda_i \left(\frac{M}{\Lambda} * r - E[X] \right)}{\sum_{i=1}^{N} \tilde{\lambda}_i} \tag{18}$$

Here *m*m_i is the number of active servers in class i , and $\tilde{\lambda}_i = \lambda_i / \delta_i$ is the normalized arrival rate. The first term of formula (18) ensures that the sub-cluster in class i will not be overloaded. The second term is related to arrival rates, differentiation parameters, and *r.*

We can also derive the expected slowdown of class i as:

$$E[S_i] = \frac{\delta_i E[X^2]E[X^{-1}] \sum_{i=1}^{N} \tilde{\lambda}_i}{2 \sum_{i=1}^{N} \lambda_i \left(\frac{M}{\Lambda} * r - E[X] \right)} \tag{19}$$

From (19) we can observe that the slowdown of class i is proportional to the pre-specified parameter δ_i , and is related to *r*. The slowdown ratio only depends on the pre-defined differentiation parameters.

The expected request time for class i can be calculated as:

$$E[R_i] = \frac{\delta_i E[X^2] \sum_{i=1}^{N} \tilde{\lambda}_i}{2 \sum_{i=1}^{N} \lambda_i \left(\frac{M}{\Lambda} * r - E[X] \right)} + E[X] \le \beta_i \tag{20}$$

β_i is request time constraint for class i . We can learn from equation (20), request time in class i is also independent of workload, but depends on both the pre-specified parameter δ_i and *r*.

The performance is controllable based on our energy proportional model, with acceptable performance degradation; large amounts of energy can be saved.

5. PERFORMANCE EVALUATION

A. Simulation Results

We build a simulator which consists of a package generator, a server dispatcher, a number of waiting queues, and a number of servers. The package generator produces incoming requests with exponential inter-arrival time distribution and bounded Pareto packet size distribution. The GNU scientific library is used for stochastic simulation.

Simulation parameters are set as follows. The shape parameter α of the bounded Pareto distribution is set to 1.5. The lower bound *k* and upper bound *p* were set to 0.1 and 100, respectively [44]. The number of servers in the cluster is 20. And we set the normalized maximum jobs one server can handle Λ/M = 1. We set the power consumption 160W for active nodes [32].

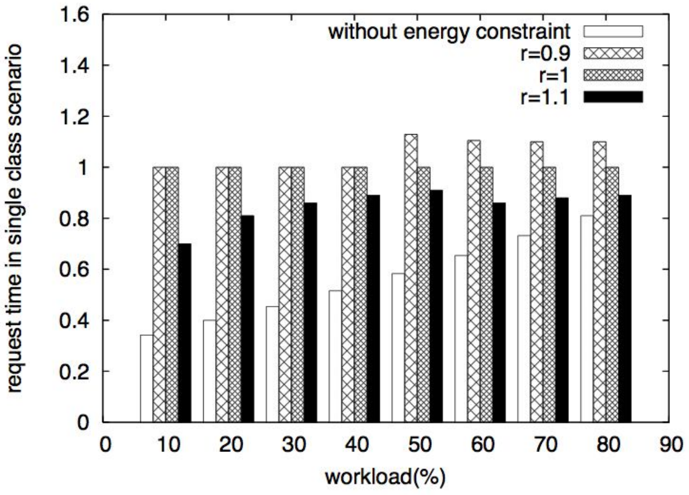

Fig. 2. Comparison of request time between non-energy proportional model and energy Proportional model. *r* is set differently according to different requirements of performance in a single class scenario.

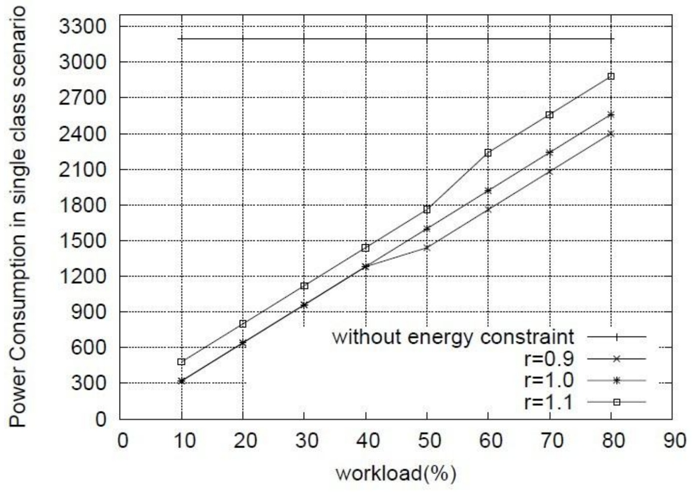

Fig. 3. Comparison of power consumption between non-energy proportional model and energy proportional model in a single class scenario. *r* is set differently according to different requirements of performance. we can achieve considerable energy saving with energy proportional model.

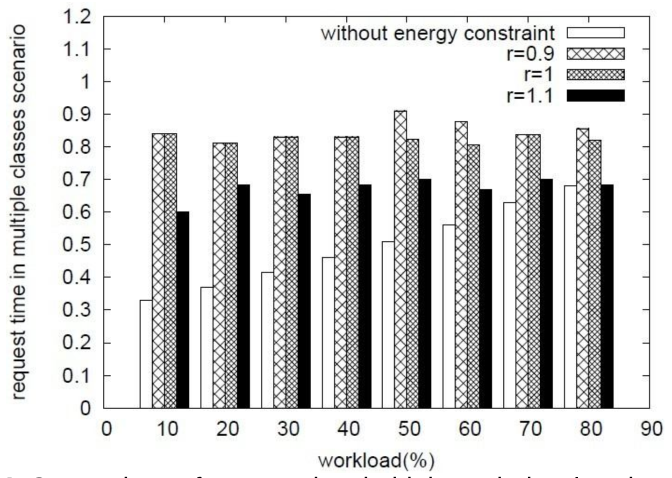

Fig. 4. Comparison of request time in higher priority class between non-energy proportional model and energy proportional models. *r* is set differently according to different requirements of performance in a multiple classes scenario.

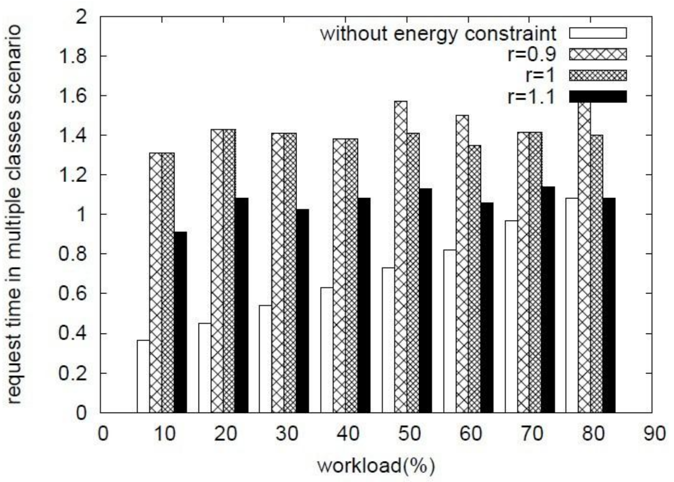

Fig. 5. Comparison of request time in lower priority class between Non-energy proportional model and energy proportional models. *r* is set differently according to different requirements of performance in a multiple classes scenario.

We first evaluate the energy proportional model for the single class scenario. We set the request time β= 0.9, β= 1.1 and β= 1.3 which correspond to adjustment parameter *r* = 1.1, *r* = 1 and *r* = 0.9 respectively. We show the simulation results in the workload range of 10% - 80%. When the workload is above 80%, the impact of energy proportionality constraint is very limited. Since the typical server operating range is between 10% - 60%, the results presented here are sufficient to test the energy proportional model.

As Figure 2 indicates that the request time is always around the pre-defined performance parameter under different workload. The request time increases as the value of *r* decreases. The results show that with adjustable parameter *r* desirable service performance can be achieved. Figure 3 compares the energy consumption of energy proportional model and non-energy proportion model for a single class scenario. We can achieve better energy efficiency under low workload, which leads to large amounts of energy saving in a server cluster.

Next, we compare the performance metrics in a multiple classes' scenario, as shown in Figure 4, 5, 6. The number of classes is normally two or three [45][46]. In this paper we choose two classes of incoming requests. We set the target slowdown ratio $\delta_2 : \delta_1$ = 2 : 1. The energy curve parameters are set differently according to different request time constraints. Note, in a multiple classes scenario, parameter r is determined by performance requirements of all classes, which means it should be set to be the largest value satisfying the requirements of all the classes. We observe that the model can achieve desirable proportionality of slowdown differentiation with request time constraints. Figure 7 also compares the energy consumptions for proportional and non-proportional models in multiple classes scenario.

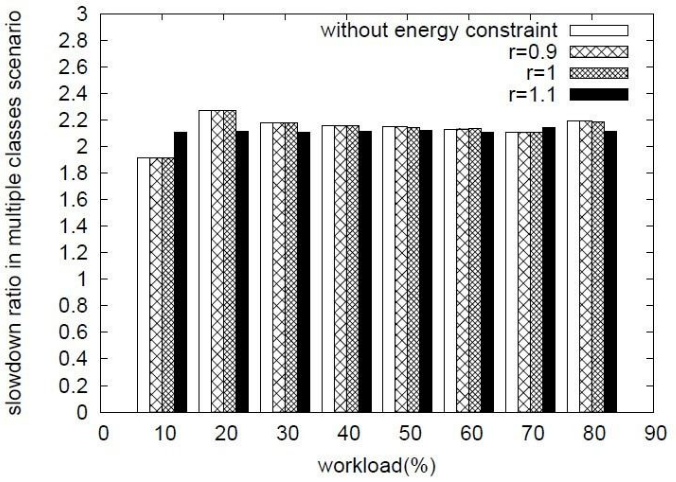

Fig. 6. Comparison of slowdown ratio between non-energy proportional model and energy proportional models. *r* is set by different requirements of performance in a multiple classes scenario.

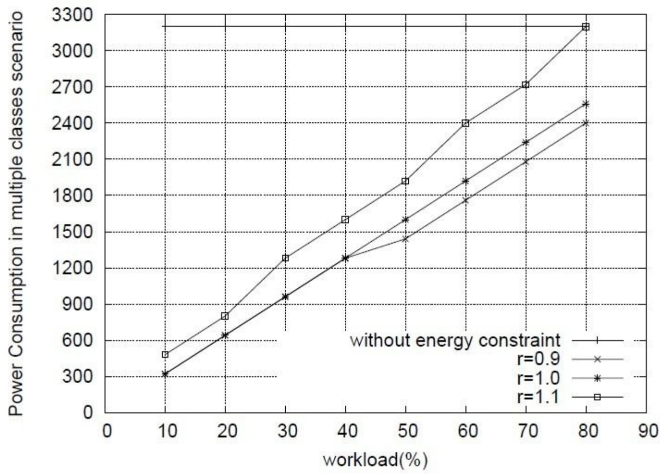

Fig. 7. Comparison of power consumption between non-energy proportional model and energy proportional model in multiple classes scenario. *r* is set by different requirements of performance. We can achieve considerable energy saving compare to the non-energy proportional model.

B. Transition Overhead Analysis

The model proposed in this paper is a continual allocation process, where we dynamically change the number of active servers. The transition time when a server transfers from an inactive mode to an active mode can not be ignored, this can influence the performance during the transition period. Thus, it is necessary to estimate the cost of transition overhead.

Generally speaking, the transition time for different servers is different which depends on the processor and other hardware constraints. Therefore, we study the influence on performance caused by transition overhead under different time. Figure 8 shows how the request time changes when considering transition overhead as the workload gradually changed from 0%-80% based on the energy proportional model. We only concern the situation when the workload increases, since as the workload decreases, the number of active servers will decline, which will not cause performance degradation. The y-axis is the request time under different transition overhead. As indicated in the figure, larger transition time has more impact on performance. The performance will be affected greatly when large number of servers can not transfer to active mode on time.

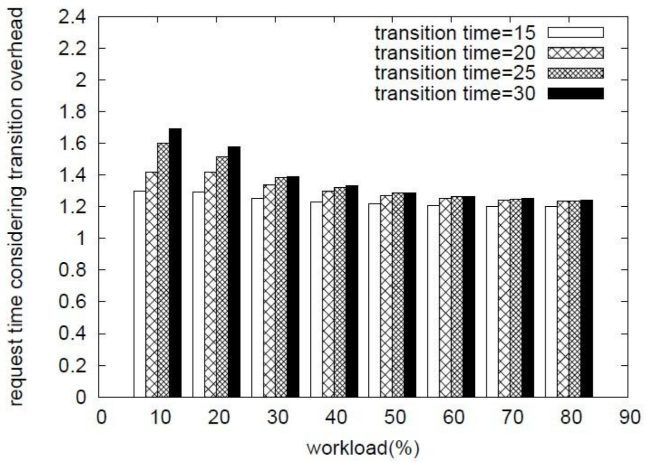

Fig. 8. The effect to performance of transition overhead in energy proportional model, the transition time is set to be 15,20,25,30 respectively

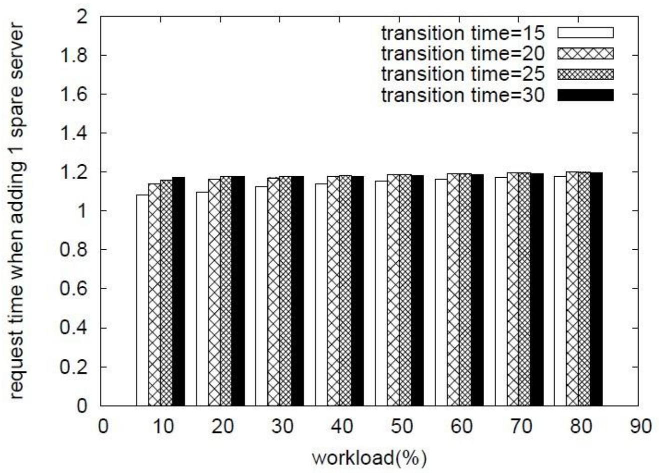

Fig. 9. Request time after adding one spare server based on energy proportional model in a single class scenario, the transition time is set to be 15,20,25,30 respectively.

It is important to make sure that the QoS is not sacrificed excessively in favor of power and energy savings. Spare servers are added to solve the problem of transition overhead. Figure 9, 10 illustrate the performance after one and two spare servers are added in a single class scenario. By adding one spare server, the performance can be improved dramatically compared to the case of no spare server. Adding two spare servers, the response time can stay under the pre-defined threshold when the workload gradually changes from 0%-80%. However, in some special situations, the workload may vary significantly within two control periods. One or two spare servers are not adequate to compensate the performance degradation. More spare servers are required.

C. Performance Evaluation Based on Real Workload Data Trace

To evaluate the model on realistic traffic patterns, we use an hour's workload trace collected by Lawrence Berkeley National Laboratory [47]. Request time threshold is set to be β= 0.6 and $r=1$. Figure 11 illustrates the performance based on our model in a single class scenario. The requests arrival rate and job size are normalized. We evaluate the performance in the situations of non-spare server and spare servers respectively. As shown in the figure, when the workload decreases, there is no performance degradation, however the performance degradation can be clearly seen as the workload increases in the case of no spare server is added. With one or two spare servers, the performance can be improved significantly. Especially, when two spare servers are always on, request time is always under predefined threshold. The result also indicates that as the number of spare server increases, the performance does not change dramatically. The request time tends to stay in a level, which demonstrates proper spare servers should be set to compensate the performance degradation.

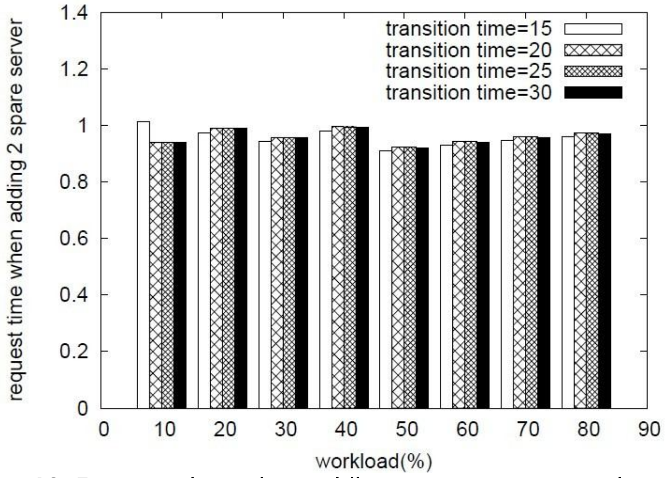

Fig. 10. Request time when adding two spare servers based on energy proportional model in a single class scenario, the transition time is set to be 15,20,25,30 respectively.

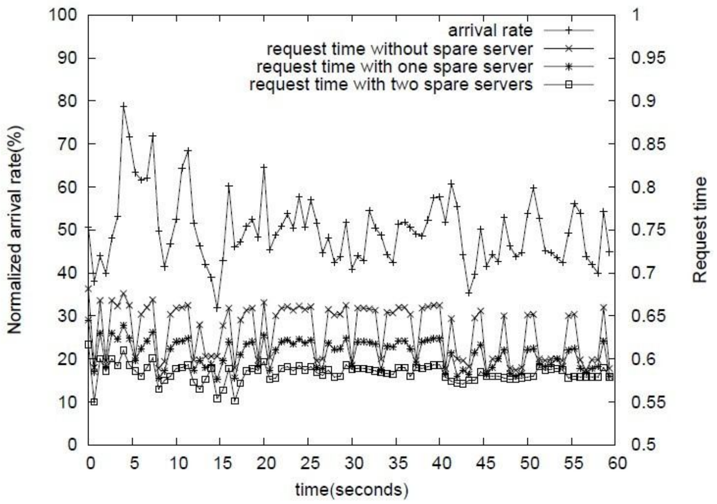

Fig. 11. Request time when adding two spare servers based on energy proportional model in a single class scenario.

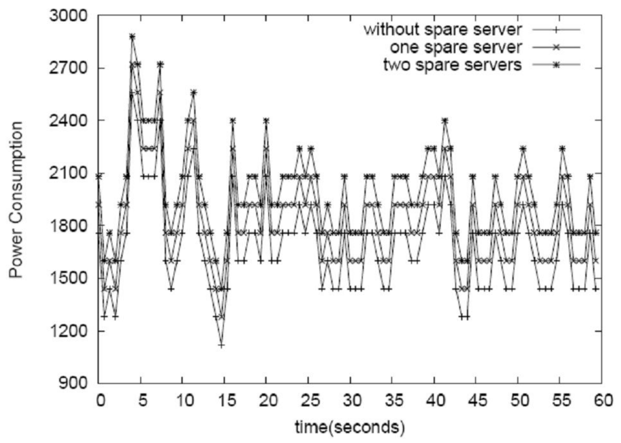

Fig. 12. Power consumption when adding two spare servers based on energy proportional model in a single class scenario.

Figure 12 evaluates the power consumption based on our model under real workload data trace. The system arrival rate is the same as shown in figure 11. The power consumption is dynamically changed as the workload changed. With little more power consumption, we can achieve better performance, and eliminate the effect of transition overhead.

6. CONCLUSION AND FUTURE WORK

Energy management becomes a key issue in server clusters and data centers. This paper aims at providing effective strategies to reduce power consumption and reduce the impact of performance degradation. We summarize out work as follows: first, the energy proportional model based on queuing theory can provide accurate, controllable and predictable quantitative control over power consumption; second, we analyze the effect of transition overhead and propose a strategy to improve the performance efficiency. Finally we evaluate the energy proportional model via simulation. Simulation results show that the energy proportional model can achieve predictable and controllable proportional energy consumption and desirable performance in a server cluster.

Future work would include studying the effect on performance when applying different dispatching strategies in our model. We are still trying to extend the server states to solve the problem of non-integer number of nodes, which will further enhance the energy efficiency. Eventually our goal is to apply our model to the real Internet web servers in the future.

7. REFERENCES

[1] U.S. Environmental Protection Agency. Report to Congress on Server and Data Center Energy Efficiency.August 2007.
[2] J. S. Aronson, “Making it a positive force in environmental change,” IT Professional, vol. 10, pp. 43 – 45, Jan 2008.
[3] US Congress. House bill 5646. To study and promote the use of energy efcient computer servers in the united states.http://www.govtrack.us/congress/bill.xpd?bill=h109-5646. Retrieved: 02-14-2008.
[4] G. von Laszewski, L. Wang, A. J. Younge, and X. He, “Poweraware scheduling of virtual machines in dvfs enabled clusters,” Cluster Computing and Workshops, 2009. CLUSTER ’09. IEEE International Conference on, pp. 1 – 10, Jan 2009.
[5] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, and Q. Wang, “Managing server energy and operational costs in hosting centers,” Proceedings of the 2005 ACM SIGMETRICS international, Jan 2005.

[6] C. Lefurgy, K. Rajamani, F. Rawson, and W. Felter, “Energy management for commercial servers,” Computer, Jan 2003.
[7] Y. Lu and G. D. Micheli, “Operating-system directed power reduction,” In proc. of international symposium on Low power electronics and design, Jan 2000.
[8] C. Lefurgy, X. Wang, and M. Ware, “Server-level power control,” Autonomic Computing, 2007. ICAC ’07. Fourth International Conference on, pp. 4 – 4, May 2007.
[9] X. Wang and M. Chen, “Cluster-level feedback power control for performance optimization,” In Proc. of Symposium on High-Performance Computer Architecture, Jan 2008.
[10] L. Barroso and U. Holzle, “The case for energy-proportional computing,” Computer, vol. 40, pp. 33 – 37, Dec 2007.
[11] G. Quan and X. Hu, “Energy efficient fixed-priority scheduling for real-time systems on variable voltage processors,” Design Automation Conference, Jan 2001.
[12] M. Elnozahy, M. Kistler, and R. Rajamony, “Energy conservation policies for web servers,” Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems, Jan 2003.
[13] J. Pouwelse, K. Langendoen, and H. Sips, “Energy priority scheduling for variable voltage processors,” Proceedings of the 2001 international symposium on Low power, Jan 2001.
[14] K. Skadron, T. Abdelzaher, and M. Stan, “Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management,” pp. 17–28, Feb. 2002.
[15] Q. Wu, P. Juang, M. Martonosi, L. Peh, and D. Clark, “Formal control techniques for power-performance management,” IEEE Micro, Jan 2005.
[16] M. Femal and V. Freeh, “Boosting data center performance through non-uniform power allocation,” Autonomic Computing, Jan 2005.
[17] R. Graybill and R. Melhem, “Power aware computing,” books.google.com, Jan 2002.
[18] D. Brooks and M. Martonosi, “Dynamic thermal management for high-performance microprocessors,” High-Performance Computer Architecture, Jan 2001.
[19] W. Felter, K. Rajamani, T. Keller, and C. Rusu, “A performanceconserving approach for reducing peak power consumption in server systems,” Proceedings of the 19th annual international conference on Supercomputing, Jan 2005.
[20] T. Newhall, D. Amato, and A. Pshenichkin, “Reliable adaptable network ram,” 2008 IEEE International Conference on Cluster Computing, Jan 2008.
[21] A. Weissel, B. Beutel, and F. Bellosa, “Cooperative I/O–a novel I/O semantics for energy-aware applications,” usenix.org.
[22] D. Helmbold, D. Long, T. Sconyers, and B. Sherrod, “Adaptive disk spindown for mobile computers,” Mobile Networks and Applications, Jan 2000.
[23] S. Gurumurthi, A. Sivasubramaniam, and M. Kandemir, “DRPM: dynamic speed control for power management in server class disks,” Computer Architecture, Jan 2003.
[24] M. Vasic, O. Garcia, J. Oliver, P. Alou, and J. Cobos, “A dvs system based on the trade-off between energy savings and execution time,” Control and Modeling for Power Electronics, 2008. COMPEL 2008. 11th Workshop on, pp. 1 – 6, Jul 2008.
[25] E. Pinheiro, R. Bianchini, E. Carrera, and T. Heath, “Dynamic cluster reconfiguration for power and performance,” Compilers and operating systems for low power, Jan 2001.
[26] V. Sharma, A. Thomas, T. Abdelzaher, K. Skadron, and Z. Lu, “Poweraware qos management in web servers,” 24th IEEE Real-Time Systems Symposium, Jan 2003.
[27] C. Dovrolis, D. Stiliadis, and P. Ramanathan, “Proportional differentiated services: Delay differentiation and packet scheduling,” Proceedings of the conference on Applications, Jan 1999.
[28] R. Sharma, C. Bash, C. Patel, and R. Friedrich, “Balance of power: Dynamic thermal management for internet data centers,” IEEE Internet Computing, Jan 2005.
[29] X. Fan, W. Weber, and L. Barroso, “Power provisioning for a warehouse-sized computer,” Proceedings of the 34th annual international conference on architecture, Jan 2007. B-2-2-2.
[30] R. Guerra, J. Leite, and G. Fohler, “Attaining soft real-time constraint and energy-efficiency in web servers,” Proceedings of the 2008 ACM symposium on Applied computing, Jan 2008.
[31] S. Nedevschi, L. Popa, G. Iannaccone, and S. Ratnasamy, “Reducing network energy consumption via sleeping and rate-adaptation,” NSDI, Jan 2008.
[32] G. Chen, W. He, J. Liu, S. Nath, L. Rigas, and L. Xiao, “Energy-aware server provisioning and load dispatching for connection-intensive internet services,” Proceedings of the 5th USENIX

Symposium on Networked Systems Design and Implementation, Jan 2008.
[33] S. Murugesan, “Harnessing green it: Principles and practices,” IT Professional, Jan 2008.
[34] M. Kesavan, A. Ranadive, A. Gavrilovska, and K. Schwan, “Active coordination (act)–toward effectively managing virtualized multicore clouds,” 2008 IEEE International Conference on Cluster Computing, Jan 2008.
[35] L. Hu, H. Jin, X. Liao, X. Xiong, and H. Liu, “Magnet: A novel scheduling policy for power reduction in cluster with virtual machines,” 2008 IEEE International Conference on Cluster Computing, Jan 2008.
[36] J. Chase, D. Anderson, P. Thakar, and A. Vahdat, “Managing energy and server resources in hosting centers,” Proceedings of the eighteenth ACM symposium on Operating Operating System Principles, Jan 2001.
[37] I. Ahmad, S. Ranka, and S. Khan, “Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy,” pp. 1–6, April 2008.
[38] E. Carrera, E. Pinheiro, and R. Bianchini, “Conserving disk energy in network servers,” Proceedings of the 17th annual international conference on Supercomputing, Jan 2003.
[39] M. Song, “Energy-aware data prefetching for multi-speed disks in video servers,” Proceedings of the 15th international conference on Supercomputing, Jan 2007.
[40] X. Zhou, Y. Cai, C. Chow, and M. Augusteijn, “Two-tier resource allocation for slowdown differentiation on server clusters,” Parallel Processing, Jan 2005.
[41] X. Zhou, Y. Cai, G. Godavari, and C. Chow, “An adaptive process allocation strategy for proportional responsiveness differentiation on web servers,” Web Services, 2004. Proceedings. IEEE International Conference on, pp. 142 – 149, Jun 2004.
[42] C. Dovrolis and P. Ramanathan, “A case for relative differentiated services and the proportionaldifferentiation model,” Network, Jan 1999.
[43] X. Zhou and C. Xu, “Harmonic proportional bandwidth allocation and scheduling for service differentiation on streaming servers,” Parallel and Distributed Systems, IEEE Transactions on, vol. 15, pp. 835 –848, Sep 2004.
[44] M. Harchol-Balter and C. U. PITTSBURGH, “Task assignment with unknown duration,” doi.ieeecomputersociety.org, Jan 1999.
[45] H. Zhu, H. Tang, and T. Yang;, “Demand-driven service differentiation in cluster-based network servers,” INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, vol. 2, pp. 679 – 688 vol.2, Mar 2001.
[46] L. Zhang, “A two-bit differentiated services architecture for the internet,” Request for Comments (Informational), Jan 1999.
[47] NASA Kennedy Space Center Server Traces. http://ita.ee.lbl.gov/html/traces.html.

Network Service Description and Discovery for the Next Generation Internet

Qiang Duan qduan@psu.edu
Information Science and Technology Department
The Pennsylvania State University Abington College
Abington, PA 19001, USA

Enyue Lu ealu@salisbury.edu
Mathematics and Computer Science Department
Salisbury University
Salisbury, MD, 21801, USA

Abstract

The next generation Internet will face new challenges due to the coexisting heterogeneous networks and highly diverse networking applications. Therefore how to coordinate heterogeneous networking systems to support a wide spectrum of application requirements becomes a significant research problem. A key to solving this problem lies in effective and flexible collaborations among heterogeneous networking systems and interactions between applications and the underlying networks. Network virtualization plays a crucial role in enabling such collaborations and interactions, and the Service-Oriented Architecture (SOA) provides a promising approach to supporting network virtualization. Network service description and discovery are key technologies for applying SOA in networking, and the current service description and discovery technologies must be evolved to meet the special requirements of future Internet. In this paper, we study the problem of network service description and discovery to support network virtualization in the next generation Internet. The main contributions of this paper include a general approach to describing service capabilities of various heterogeneous networking systems, a technology to discover and select the network services that guarantee the QoS requirements of different networking applications, a general profile for specifying networking demands of various applications, a scheme of network resource allocation for QoS provisioning, and a system structure for realizing the network description, discovery, and resource allocation technologies. We also propose information update mechanisms for improving performance of the network service description and discovery system. The approach and technology developed in this paper are general and independent of network architectures and implementations; thus are applicable to the heterogeneous networking systems in the next generation Internet.

Keywords: SOA, Internet, network virtualization, service description, service discovery, QoS.

1. INTRODUCTION

Although became a global communication platform in a short period of time, the Internet has fallen victim to its own stunning success. The next generation Internet will face many challenges, which essentially come from the diversity in the networking architectures and technologies coexisting in the Internet and the wide spectrum of networking applications supported by the Internet. The numerous distributed computing applications with diverse networking requirements motive research on alternative networking technologies and architectures for the future Internet. The newly developed networking technologies vary in almost all aspects of network functionalities, including data forwarding mechanisms, routing and signaling protocols, and control and management schemes. So far no single network architecture has demonstrated the capability of supporting all kinds of networking applications effectively and efficiently. The coexistence of various heterogeneous networking systems will be one of the essential features of the next generation Internet. Recently the networking research community started exploring "clean-slate" approaches to develop new Internet architectures, and an important aspect of the objective is to enable alternative network architectures coexist and collaborate inside future Internet. Therefore, how to coordinate heterogeneous networking systems to support the wide variety of application requirements becomes a significant research problem.

A key to solve this problem lies in flexible collaboration across heterogeneous networking systems and effective interactions between networks and applications. Recent research progresses toward this direction include new network architecture for diversifying the Internet [23], a concurrent architecture (CABO) that separates the roles of service providers and network infrastructure providers [10], a new network control plane developed in the DRAGON project [17], and the network composition mechanism for Ambient Networks [6]. Examples of current standardization efforts for supporting various applications across heterogeneous networks include the ITU-T Next Generation Network (NGN) structure [28] and the Open Service Environment (OSE) defined by the Open Mobile Alliance (OMA) [29]. Although the above researches address the heterogeneous networking problem from different aspects, they share a similarity in which the notion of *network virtualization* plays a crucial role. Essentially network virtualization is to abstract networking resources into reusable components that can be discovered, selected, and composed to meet different networking requirements. Therefore an effective mechanism for network virtualization will greatly facilitate the next generation Internet.

The *Service-Oriented Architecture* (SOA) [25] is currently gaining high attention and acceptance in IT industry, especially in the fields of Web services and Grid/Cloud Computing. In SOA, heterogeneous computational resources are virtualized into *services* that are discoverable, selectable, and composable for supporting various computing application requirements. A key feature of SOA is the loose-coupling mechanism that enables flexible and effective interactions among services and applications. Therefore applying ideas of SOA in networking will greatly facilitate network virtualization in the next generation Internet. Following the SOA principles, heterogeneous networking systems can be virtualized into *network services* which are reusable network components whose functions and access methods are described in standard-format documents called network service descriptions. The network service descriptions can be published at a service registry. When an application needs to utilize networking resources, it sends a request to a network service broker, which will discover and select the appropriate network service that meets the application requirements. Therefore network service description and discovery form the foundation of this service-oriented networking paradigm.

Although SOA has been successfully applied in the field of distributed computing, currently available technologies for service description and discovery must to be enhanced to meet the special requirements of the next generation networking. One of the key requirements is to support the network Quality of Service (QoS) required by applications. The current service description standard in SOA mainly focuses on functional information instead of service provisioning capabilities; thus limiting service discovery to be function-based instead of performance-based. Although progresses have been made toward enabling performance-based Web and Grid services discovery, the obtained results may not be applied directly to networking systems. Therefore, new approaches to describing network QoS capability and discovering network services based on their achievable QoS must be developed.

In this paper we address the problem of service-oriented network description and discovery to meet the requirements of network virtualization in the next generation Internet. The heterogeneity of networking systems and the diversity of application requirements in future Internet make this problem challenging. The main contribution made by this paper include a general approach to describing data delivery capabilities of various heterogeneous networking systems, a technology to discover and select the network services that can guarantee the QoS required by different networking applications, a general profile for specifying the networking demands of various applications, a scheme of network resource allocation for QoS provisioning, and a system structure for realizing the developed network description, discovery, and resource allocation technologies. We also propose information update mechanisms for improving the scalability and performance of the network service description and discovery system.

The rest of this paper is organized as follows. Section 2 reviews networking for the next generation Internet and discusses the application of SOA in this area. Section 3 proposes a new approach for describing network service capabilities. Section 4 develops the technology for network service discovery and resource allocation for network service provisioning. Numerical examples are provided in Section 5 to illustrate applications of the developed technologies. In Section 6 we design a system structure for network service description and discovery, and discuss scalable mechanisms for updating network service information. Section 7 draws conclusions.

2. SERVICE-ORIENTED NETWORKING FOR THE NEXT GENERATON INTERNET

2.1 Networking for the Next Generation Internet

The next generation Internet will be featured by a wide spectrum of supported networking applications and the diverse coexisting network infrastructures. The diversity of network infrastructures referenced here includes the type of networking technologies, service capabilities and provisioning mechanisms, administrative ownership, network management and control policies, among others. The coexisting diverse networking systems must cooperate with each other in order to deliver end-to-end network services for supporting various network applications. The current Internet architecture lacks the flexibility to face this challenge, therefore developing new network architectures for enabling flexible interactions among heterogeneous networks to support various applications has become an active research area.

Recently a new network architecture was proposed for diversifying the Internet [23]. This architecture enables various meta-networks built on top of a shared substrate comprising heterogeneous networking resources. Such an architecture allows network providers to automatically deploy, configure, and operate meta-networks to meet application requirements. A similar idea was also developed in the CABO architecture ("Concurrent Architecture are Better Than One") [10] for the future Internet. The CABO architecture decouples infrastructure providers (who manage the physical infrastructure) and network service providers (who deploy network protocols and offer end-to-end services); thus supporting multiple simultaneous network architectures on top of shared physical infrastructures. An inter-domain network control plane was developed in the DRAGON (Dynamic Resource Allocation in GMPLS Optical Networks) project [17] for enabling dynamic provisioning of networking resources for high-performance Grid and e-science applications. A crucial component in this control plane is the Network-Aware Resource Broker (NARB) that represents the local autonomous systems. A networking paradigm called Ambient Network (AN) [6] was developed to support ubiquitous provisioning of network services over any type of network. A core feature of Ambient Network is a uniform mechanism called *network composition* for dynamic and scalable cooperation between heterogeneous networks. A service plane architecture for multi-domain network connections was also reported in [12], which introduced the notion of inter-domain network service as the result of composing a set of service elements. Some network standard organizations are also working on supporting applications across heterogeneous networks. The Open Mobile Alliance (OMA) recently developed an Open Service Environment (OSE) that delivers network services by composing a set of standard service components called service enablers [29]. The Next Generation Network (NGN) structure defined by ITU-T promotes separation between service delivery from network transport infrastructure [28].

Although the above mentioned research and standardization efforts tackle the problem of coordinating heterogeneous networking systems to support various applications from different aspects, the notion of network virtualization plays a significant role in all these proposed solutions. Through network virtualization, the underlying networking resources are de-coupled from network service provisioning. Networking resources are encapsulated into reusable components, which can be discovered, selected, and composed to meet application requirements. In the new Internet architecture proposed in [23], a meta-network is a virtual network that serves as an abstraction of a collection of heterogeneous networking resources. The CABO architecture employs network virtualization as a key mechanism to enable service providers share the underlying network infrastructures for end-to-end service delivery. In the DRAGON network control plane, a key technology for realizing NARB is network domain virtualization, which provides a mechanism to advertise a simplified view of a network domain. Network composition in Ambient Networks is based on a connectivity and resource virtualization framework that hides the differences of heterogeneous networking systems and enables applications to operate across them. The service element proposed in [12] is also a type of virtualization of networking resources, which can be discovered and composed to form inter-domain network services. The service enablers in the OMA OSE are reusable virtualizations of networking resources that provide access to network capabilities. The separation between service provisioning and network transportation in the NGN structure is also realized through network virtualization. Therefore, an effective and flexible mechanism for network virtualization will greatly facilitate the development of new networking architectures and technologies for the next generation Internet.

2.2 The Service-Oriented Architecture and Its Application in Networking

The *Service-Oriented Architecture* (SOA) is a system architecture initially developed by the IT community, especially in the areas of Web services and Grid/Cloud computing, as an effective solution to coordinating computing resources crossing heterogeneous systems to support various applications. The SOA is described as “an architecture within which all functions are defined as independent services with invokable interfaces that can be called in defined sequences to form business processes” [9]. Services in SOA are self-contained and reusable computing components that can cooperate with other services through pre-defined standard interfaces. Essentially the SOA enables virtualization of various computing resources in form of services and provides a flexible interaction mechanism among services.

The interaction mechanism in the SOA is briefly illustrated in Figure 1. A service provider publishes a machine-readable document called *service description* at a service registry. The service description gives descriptive information about the functions provided by the service and the interfaces for utilizing such functions. When a service customer, either an application or another service, needs to utilize computing resources to perform a certain function, it starts a service discovery process to locate an available service that meets its requirement. Typically a *service broker* handles service discovery for service customers by searching the service descriptions published at the registry and selecting a service that matches the criteria specified by the customer. After discovering a service, the service customer contacts the service provider and invokes the service by following the interface defined in the service description. Currently the

SOA is realized by a set of Web Services standards, including Web Service Description Language (WSDL) [35], Universal Description Discovery and Integration (UDDI) [32], Simple Object Access Protocol (SOAP) [36], which are all based on Extensible Markup Language (XML).

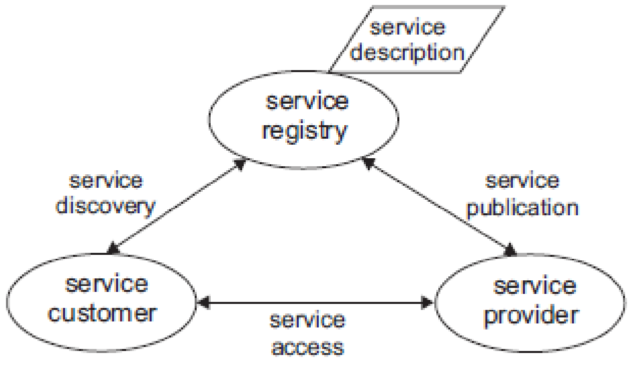

FIGURE 1: The service-oriented architecture.

A key feature of SOA is the “loose-coupling” interactions among heterogeneous systems in the architecture, including service providers, service customers, and the service broker and registry. “Loose-coupling” means entities can effectively interact with each other while keep themselves independent. It is this feature makes the SOA a very effective architecture for coordinating heterogeneous systems to support various application requirements, which is essentially the same challenge faced by the next generation Internet. Therefore, applying the SOA principles in the field of networking provides a promising approach to constructing the next generation Internet. In this paper we refer to such SOA-based networking paradigm as *Service-Oriented Networking*. In this paradigm network virtualization can be realized through encapsulating networking resources into *network services*. A network service may represent any type of networking component, could be a network domain, a collection of networks, a single physical network, or just a network node. Multiple network services can be composed into one composite inter-network service. By publishing a network service description, each network service can advertise its functions and capabilities without exposing internal implementation details.

The service-oriented networking paradigm is shown in Figure 2. In this paradigm a network service provisioning layer is deployed between networking applications and the underlying networking platform. The networking platform consists of heterogeneous networking resources encapsulated in network services. When an application needs to utilize networking resources, it sends a request to a network service broker. The network broker searches the network service descriptions published at the service registry and selects the network service that can meet the application requirement. After receiving a response from the service broker with a network selection result, the application may start accessing the networking resources provided by the selected network service.

SOA-based network virtualization gives Internet service providers and network operators the ability to view their underlying network infrastructure more as a commodity and allows infrastructure development to become more consistent. SOA-based network service provisioning also enables faster time to market as new initiatives can reuse existing services and components, thus reducing design, development, testing, and deployment time in addition to the cost and risk of undertaking such projects.

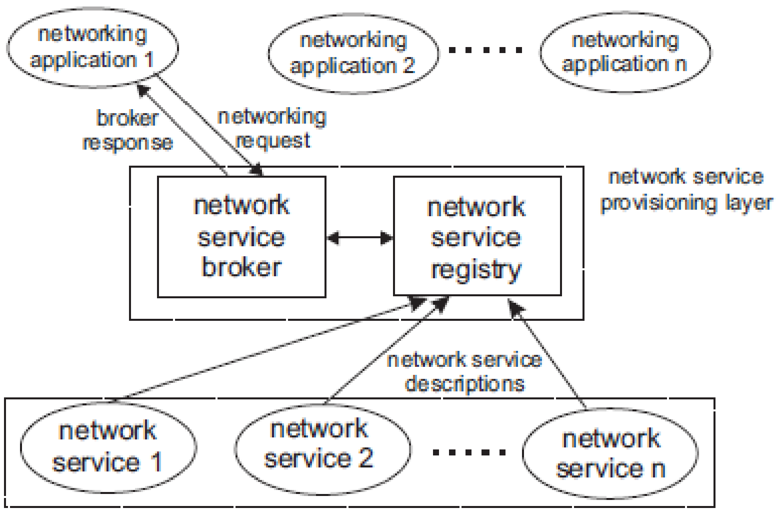

FIGURE 2: Network service description and discovery in the service-oriented networking paradigm.

2.3 Network Service Description and Discovery

Network service description and discovery are key components for adopting SOA in networking. Service-oriented network virtualization introduces new challenges to description and discovery of network services. A distinguishing feature for network services is their service provisioning capabilities, namely the capability of guaranteeing a certain level of QoS performance to an application. Therefore, service description for network virtualization should provide information about network service provisioning capability. Since most applications for the next generation Internet require high-performance networking, the key to network service discovery is selecting the appropriate network services that meet application performance requirements; that is, performance-based network service selection. The currently available service description and discovery technologies in SOA must be enhanced to meet these new requirements.

The current standards for service description and discovery are Web Service Description Language (WSDL) [35] and Universal Description, Discovery and Integration (UDDI) [32]. The WSDL specification defines an abstract interface describing the functionality of a service and a concrete interface that describes how to contact and invoke the service. The UDDI specification defines the interfaces for publishing and searching service descriptions and the data structures for organizing service description information. Currently the WSDL specification focuses on providing functional information about services and UDDI lacks effective mechanism to publish and search non-functional features such as service provisioning capability. Therefore, the service description and discovery technologies based on WSDL and UDDI are function-based instead of performance-based.

Research efforts have been made for enabling QoS-capable service description, discovery, and selection. For example, R.Al-Ali and his colleagues developed an extended UDDI registry (UDDIe) and the G-QoSM framework for supporting QoS-based Grid service discovery [1, 2]. However, the UDDIe was mainly tied with the G-QoSM framework for Grid computing and had limited support for QoS details. A QoS-capable service broker algorithm was developed to discover, select, and compose Web services that meet end-to-end QoS constraints [26, 27]. Al-Masri and Mahmoud introduced a relevancy ranking function based on QoS parameters to find the best Web service that meet client QoS preferences [3]. Technologies for QoS-aware run-time service discovery and selection were proposed in [4, 24] and a model to filter discovered services with their QoS features to maximize user satisfaction was developed in [18]. These technologies were mainly developed for Web or Grid services focusing on data processing and computing instead of data communications and networking. Therefore, the obtained results may not be applied directly to network service description and discovery.

World Wide Web Consortium (W3C) recently offered WS-Policy [37] and WS-PolicyAttachment [38] specifications for standardizing the description of non-functional characteristics of Web services. WS-Policy aims to provide a general-purpose framework and model for expressing non-functional service characteristics such as requirements, preference, and capabilities as policies. WS-PolicyAttachment defines a mechanism to associate the policy expressions with the existing WSDL standard. The WS-Agreement specification [31] developed by the Open Grid Forum (OGF) defines a protocol between service providers and users for establishing and managing service level agreements. These specifications have made significantly progresses toward QoS-enable Web/Grid service description and discovery. However, modeling and describing network service provisioning capability are left opened as domain-specific issues in these specifications.

Research results on network modeling and description have also been reported. A network modeling tool called Effective Network Views was developed in [21]. The application of this tool is mainly limited to local area networks and it is not scalable to the Internet. S. Lacour and his colleagues employed the directed acyclic graph (DAG) to describe network topology and developed a scalable network description model [16]. This description model focuses on a functional view of network topology instead of service provisioning capabilities; thus lacks the information needed for performance-based network service discovery. A Network Description Language (NDL) was developed in [14] as a semantic schema for describing network topology. The NDL language serves more as a vocabulary to present network topology than an approach to modeling network service capabilities, and the reported application of NDL mainly focused on optical networks. However the next generation Internet may consist of a wide variety of networks with different implementations.

To the best of our knowledge little work has been reported with regard to network capability description and performance-based network discovery technologies that are applicable to the heterogeneous networks in the next generation Internet. The research presented in this article addresses this problem by developing a general approach for describing network service capabilities and a technology for discovering network services that meet the QoS performance required by different networking applications.

3. NETWORK SERVICE DESCRIPTION

In this section, we will develop a new approach to describing network service capabilities for supporting the service-oriented network virtualization. The main challenge to network service description lies in the heterogeneity of the networking systems that will be coexisting in the Internet. Therefore, the network service description approach must be (a) general so as to be applicable to various network implementations and (b) flexible so as to support composing multiple heterogeneous networks into one cross-domain network service. A key for network service description is to provide sufficient amount of information about the data delivery capability of a network without exposing its implementation details. In general, network data delivery capability includes two aspects: the connectivity of the network service, which can be described by enumerating the pairs of sources and destinations between which the network transports data; and the capability of data transportation between each pair of source-destination

In order to provide a formal description for network service capabilities, we define the Capability Matrix **C**. Given a network service S with m ingress ports and n egress ports, the capability matrix **C** is an $m\times n$ matrix

$$C = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,n} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,n} \\ \cdots & \cdots & \cdots & \cdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,n} \end{pmatrix} \tag{1}$$

where $C_{i,j}$ is defined as

$$C_{i,j} = \begin{cases} 0 & \text{if no route exists from } i \text{ to } j \\ \mathrm{Q}_{\mathrm{i,j}} & \text{if a route } \mathrm{R}_{\mathrm{i,j}} \text{exists from } i \text{ to } j \end{cases} \tag{2}$$

and $Q_{i,j}$ is called the QoS descriptor for the route $R_{i,j}$, which will be further developed in the rest of this section.
According to the definitions given in (1) and (2), the capability matrix element $C_{ij} = 0$ if the network service cannot reach the egress *j* from the ingress *i*. That is, all non-zero elements in the matrix **C** describes the connectivity of the network services. If the network service provides a route from *i* to *j*, then the transportation capability of this route is described by the descriptor $Q_{i,j}$.

We adopt the notion of service curve from network calculus theory [7] to design a general QoS descriptor that is applicable to various network implementations. The network calculus theory has evolved to an effective tool for network modeling and analysis. The service curve is defined as follows in network calculus. Let $T^{in}(t)$ and $T^{out}(t)$ respectively be the accumulated amount of traffic of a flow that arrives at and departs from a server by time *t*. Given a non-negative, non-decreasing function, $S(\cdot)$, where $S(0) = 0$, we say that the server guarantees a service curve $S(\cdot)$ for the flow, if for any $t \geq 0$ in the busy period of the server,

$$T^{out}(t) \geq T^{in}(t) \otimes S(t) \tag{3}$$

where $\otimes$ denotes the min-plus convolution operation defined in network calculus as $h(t) \otimes x(t) = \inf_{s:0\leq s\leq t}\{h(t-s) + x(s)\}$. Essentially a service curve gives the minimum amount of service offered by the server to a client in an arbitrary time interval within a busy period. Therefore a service curve describes the lower bound of the service provisioning capability offered to a client.

A typical server model for networking systems is the Latency-Rate (LR) server [22], which guarantees each flow a service curve $\beta_{r,\Theta}(t) = r(t - \theta)$, where θ and r are respectively called the latency and service rate for the flow. LR server is particularly interesting to us because many packet schedulers such as Weighted Fair Queuing (WFQ) and Weighted Round-Robin (WRR), which are widely deployed in practical networking equipments, belong to this server category.

In our service description approach, we adopt the service curve guaranteed by the route $R_{i,j}$ as the QoS descriptor $Q_{i,j}$ in the matrix **C**. Since a service curve is a general data structure that is independent with network implementations, it is flexible enough to describe various networking systems. In a network where a route $R_{i,j}$ can be modeled by a LR server with a service curve $r_{i,j}(t - \theta_{i,j})$, the matrix element $C_{i,j}$ can be represented by a data structure $[r_{i,j}, \theta_{i,j}]$. Currently there are various mechanisms available for measuring and managing network state information, for example the technologies reported in [13,15, 20], which could be used to obtain the data for constructing service curves and building the matrix **C**. The methods of collecting network state information are implementation dependent and may vary in different networks, but the matrix **C** provides all network services with a general and standard approach to describing their service provisioning capabilities.

An end-to-end Internet connection typical crosses multiple networks, each of which can be virtualized as a network service. Therefore, how to compose the QoS capabilities of a set of heterogeneous links into one descriptor for the end-to-end connection is an important and challenging problem. The service curve-based description approach supports QoS descriptor composition. Known from network calculus, the service curve *S(t)* guaranteed by a series of tandem servers G_1, G_2, …, G_n, which respectively guarantees the service curves $S_1(t)$, $S_2(t)$, …, $S_n(t)$ to a flow, can be obtained through the convolution of all the service curves; that is,

$$S(t) = S_1(t) \otimes S_2(t) \cdots \otimes S_n(t) \tag{4}$$

Therefore, the QoS descriptor of the end-to-end route can be obtained from the convolution of the QoS descriptors of the links provided by all single network services.

Since typical networking systems can be modeled as LR servers, we are particularly interested in composing LR servers. Suppose each network server S_i, $i = 1, 2, \ldots, n$, guarantees a service curve $\beta_{ri,\ \theta i} = r_i\ (t - \theta_i)$, it can be proved that the convolution of these service curves is

$$\beta_{r_1,\theta_1}(t) \otimes \cdots \otimes \beta_{r_n,\theta_n}(t) = \beta_{r,\theta_\Sigma}(t) \tag{5}$$

where

$$r = \min\{r_1, r_2, \cdots r_n\} \quad \text{and} \quad \theta_\Sigma = \sum_{i=1}^{n} \theta_i \tag{6}$$

Equations (5) and (6) imply that if each link on an end-to-end network route can be described by a latency-rate server, then the end-to-end route also guarantees a latency-rate service curve, whose latency parameter is equal to the summation of all link latency parameters and the service rate parameter is limited by the link with the least service rate.

In this section we developed a general description for network service capabilities. Due to the network calculus technique employed in our development, this description approach is independent with network implementations; thus applicable to various heterogeneous networking systems in the Internet. This approach can also easily support capability description for composite network services.

4. NETWORK SERVICE DISCOVERY

In this section we develop a new technology for performance-based discovery of network services, which enables the network service broker to discover network services that guarantee QoS performance required by networking applications. This newly developed technology focus on network service selection while other components of the discovery procedure, including publishing service descriptions, searching the registry for available services, negotiating service level agreement, and binding the selected service with the application, can be implemented based on current SOA service discovery standards.

4.1 Performance Prediction for Network Service Discovery

Three aspects of information are needed by a network service broker for performance-based network service discovery for an application: (a) the provisioning capabilities of available network services; (b) the performance requirement of the application; and (c) the characteristic of network traffic generated by the application. The information (a) can be obtained from the capability matrix **C** published by the network service provider. The other two aspects of information (b) and (c), which specify the demand of a networking application, should be provided to the service broker by the application as part of its request.

Due to the large number of networking applications with various requirements, it is very important to have a common approach for describing networking demands. In this paper we define a *Demand Profile* ***P (d, L, a)*** as a general specification of application requirements. This profile consists of three elements: an address set ***d***; a traffic load descriptor ***L***; and a performance requirement set ***a***. The address set ***d*** specifies the addresses of the source and destination of data transportation required by the application. If the application only needs unicast (point to point) data delivery, the address set will consist of a pair of network addresses. If multicast (point to multi-points or multi-points to multi-points) communication is required, the set ***d*** may include multiple address pairs. The set ***a*** consists of the performance parameters required by the application. Different parameters may be included in ***a*** for different applications, but the minimum bandwidth b_{req} and the maximum delay d_{req} for data transportation are typical requirements; i.e. typically ***a*** = {b_{req} , d_{req}}. The descriptor ***L*** is used to characterize the network traffic that the application will load on a network service.

In order to describe application traffic loads in a general form, we employ the *arrival curve* as the traffic descriptor ***L***. Arrival curve is another important concept in the network calculus theory. Let $T^{in}(t)$ denote the accumulated amount of traffic generated from an application by time *t*. Given a non-decreasing, non-negative function, *A(•)*, the application is said to have an *arrival curve A(•)* if for any nonnegative *t* and *s*

$$T^{in}(t) - T^{in}(t) \leq A(t-s)\,. \tag{7}$$

Essentially the arrival curve of an application gives an upper bound for the amount of traffic that the application loads on a network service.

Currently most QoS-capable networks apply traffic regulation mechanisms at network boundaries to shape arrival traffic from applications. The traffic regulators most commonly used in practice are leaky buckets. A traffic flow constrained by a leaky bucket has an arrival curve

$$A(t) = \min\{Pt, \sigma + \rho t\},$$

where *P*, *ρ*, and *σ* are respectively the peak rate, the sustained rate, and the maximal burst size of this flow.

Now we develop a technique to predict the performance that can be guaranteed by a network service to a networking request. Among various performance requirements, in this paper we focus on the minimum bandwidth and the maximum delay for data transportation, which are important performances required by most high-performance networking applications. Network calculus provides us with an effective approach for analyzing the minimum bandwidth and maximum delay performances guaranteed by a network service. A service curve itself is a description of the minimum service capacity offered by a network, which essentially gives the minimum bandwidth guaranteed by the network to an application. Therefore, given the QoS descriptor for a route $R_{i,j}$, which is described by a service curve $S_{i,j}(t)$, the minimum bandwidth guaranteed by this route can be determined as

$$b_{\min} = \lim_{t\to\infty} \lfloor S_{ij}(t)/t \rfloor. \tag{8}$$

Suppose the traffic load of a networking request is described by an arrival curve *A(t)*, then the maximum delay d_{max} guaranteed by the network to this request can be determined as,

$$d_{\max} = \max_{t\geq 0}\{\min\{\delta : \delta \geq 0, A(t) \leq S(t+\delta)\}\}. \tag{9}$$

Since the LR server is a typical network server model and the leaky bucket is a typical traffic regulator, we specifically give the performance analysis for a network route modeled by a LR server under traffic load constrained by a leaky bucket regulator. Suppose the load descriptor of an application is *A(t)* = min{*Pt*, *σ* + *ρt*}, and the QoS descriptor of the route provided by a network to this application is *Q* = [*r, θ*], then the minimum bandwidth that can be guaranteed by the route is

$$b_{\min} = \lim_{t\to\infty} \frac{r(t-\theta)}{t} = \lim_{t\to\infty}\left(r - \frac{r\theta}{t}\right) = r\,. \tag{10}$$

By following (9) we can get that the maximum delay guaranteed to the application is

$$d_{\max} = \theta + \left(\frac{p-r}{p-\rho}\right)\frac{\sigma}{r} \text{ for } r \geq \rho. \tag{11}$$

After determining the achievable performance of a network service *S* for an networking request *R*, the network broker compares the predicted performance with the requirement given in the set ***a*** of the demand profile ***P*** to decide if *S* can be selected for *R*. Networking requests can be classified into three categories according to their networking performance requirements: (a) with only bandwidth requirement; i.e. ***a***=b_{req}; (b) with only delay requirement; i.e. ***a***=d_{req}; and (c) with both bandwidth and delay requirements; i.e. ***a***={ b_{req}, d_{req}}. If *R* belongs to category (a), then *S* can be selected only when $b_{min} \geq b_{req}$. If *R* belongs to category (b), then *S* can be selected only when $d_{max} \leq d_{req}$. If *R* belongs to category (c), then *S* can selected only when $b_{min} \geq b_{req}$ and $d_{max} \leq d_{req}$. If there are multiple network services meet the performance requirements, selection among them may be based on other criteria such as service cost or load balance.

4.2 Resource Allocation for Network QoS Provisioning

The selected network service should allocate sufficient amount of networking resources in order to actually guarantee network QoS provisioning. The main resource in a network service is bandwidth, therefore in this subsection we give a discussion on bandwidth allocation for QoS provisioning in network services.

Equation (11) shows that given the traffic parameters (P, ρ, σ) of a networking request, the achievable delay upper bound d_{max} is a function of the available bandwidth r. This implies that the required delay performance can be guaranteed by allocating sufficient amount of bandwidth. Equation (11) also shows that the minimum possible delay $D_{\mathrm{min}}= \theta$ when $r = P$; that is when the allocated bandwidth is equal to the traffic peak rate. Although achieving optimal delay performance, allocating bandwidth according to the peak rate causes low resource utilization, especially for applications with fluctuating load. We can also see from (11) that d_{max} is upper bounded only if $r \leq \rho$; that is, the allocated bandwidth should be at least the sustain rate ρ of the traffic load in order to achieve any delay performance guarantee. Therefore, a reasonable bandwidth allocation scheme is to determine the minimum bandwidth r_{a} ($\rho \leq r_{\mathrm{a}} \leq P$) that is sufficient to guarantee the required delay d_{req} given by a networking request.

Analysis on (11) shows that d_{max} is a decreasing function of r that achieves the maximum value $D_{\mathrm{max}} = \theta + \sigma / \rho$ when $r = \rho$. This implies that if $r > \rho$, the application will be guaranteed a delay upper bound that is less than D_{max}. Given the delay requirement d_{req}, the network service must guarantee a delay upper bound no greater than d_{req}; that is,

$$d_{\max} = \theta + \left(\frac{P-r}{P-\rho}\right)\frac{\sigma}{r} \leq d_{req}. \qquad (12)$$

Therefore, the minimum bandwidth that must be allocated for meeting (\ref{delay2}) is

$$r_a = \frac{P\sigma}{(P-\rho)(d_{req}-\theta)+\sigma}. \qquad (13)$$

Equation (13) implies that r_a is always less than the peak rate P. As traffic becomes more smooth; that is the sustain rate ρ gets closer to the peak rate P, the required bandwidth r_a approaches P.

In summary, the bandwidth allocation for guaranteeing a delay requirement d_{req} can be determined as

$$r_a = \begin{cases} \rho & d_{req} \geq D_{\max} \\ \dfrac{P\sigma}{(P-\rho)(d_{req}-\theta)+\sigma} & D_{\min} \leq d_{req} \leq D_{\max} \\ N/A & d_{req} < D_{\min} \end{cases} \qquad (14)$$

where $D_{\mathrm{max}} = \theta + \sigma / \rho$ and $D_{\mathrm{min}}= \theta$.

If the networking request has both delay and bandwidth requirements; i.e. ***a***={b_{req}, d_{req}}, then the minimum amount of bandwidth that must be allocated for this request will be

$$b_{\min} = \max\{r_a, b_{req}\}. \qquad (15)$$

Specifically we analyze bandwidth allocation for network services that can be modeled by the weighted fair queuing (WFQ) server [19], which is widely deployed in practical networking systems. It is known from [22] that for a WFQ the latency parameter for a traffic flow is θ = L(1/R

+ 1/*r*), where *L* is the maximum packet length of this flow, *R* is the total transmission capacity of the network route, and *r* is the bandwidth available to the flow on this route. For a networking application that generates a leaky bucket constrained flow with traffic parameters (P, ρ, σ), if the QoS descriptor of the route provided by a network service for the application is S=[r, θ], then we can predict that the maximum delay guaranteed to the application is

$$d_{\max} = L\left(\frac{1}{R}+\frac{1}{r}\right)+\left(\frac{P-r}{P-\rho}\right)\frac{\sigma}{r}. \tag{16}$$

Given a delay requirement d_{req}, $D_{\min} \le d_{req} \le D_{\max}$, the bandwidth allocation requirement can be determined as

$$r_a = \frac{P\sigma + L(P-\rho)}{(P-\rho)(d_{req} - L/R)+\sigma}. \tag{17}$$

5. NUMERICAL EXAMPLES

In this section, we use numerical examples to illustrate the applications of the performance prediction and bandwidth allocation techniques developed for network service discovery. We considered two networking applications utilizing a network service in our examples. The application A_1 uses the network to deliver a stream of video packets and loads the network with a flow f_1. The application A_2 transmits a flow of audio packets f_2 through the network. Both A_1 and A_2 require a small maximum packet transmission delay. We adopt the traffic parameters given in [11] for f_1; that is, the peak rate P=5.3 Mb/s, the sustained rate ρ =1.5 Mb/s, and the maximum burst size σ =140 kbits. The traffic parameters for f_2 are given in [8]; that is, the peak rate P=3.2 Mb/s, the sustained rate ρ =1.1 Mb/s, and the maximum burst size σ =300 kbits. We assume that the maximum transmission unit (MTU) of the network is L=1K bytes, and the total link capacity of the route is R=1Gb/s.

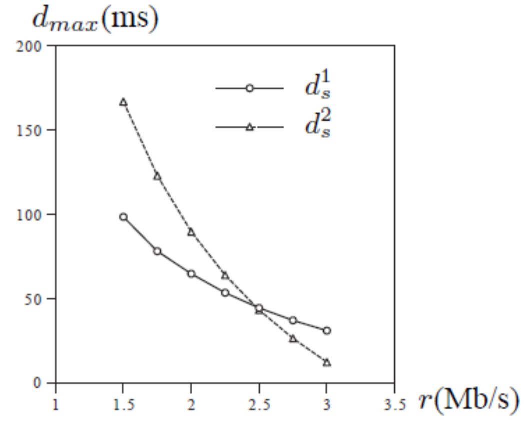

FIGURE 3: Delay performance prediction for a single network service

We first analyzed the maximum packet delay d_{max} that can be guaranteed for the flows f_1 and f_2 by a single network service with various amounts of available bandwidth *r* on the assigned network routes. The results are plotted in Figure 3, where d_s^1 and d_s^2 denote the maximum delay for f_1 and f_2 respectively. From this figure we can see that both d_s^1 and d_s^2 decrease when the available bandwidth *r* increases. This means that the more bandwidth is available to an application, the tighter is the delay upper bound guaranteed to the application. Comparison between the curves of d_s^1 and d_s^2 shows that although both of them are decreasing functions of *r* d_s^2 drops faster than d_s^1, which means that the same amount of bandwidth increment can make more significant improvement in delay performance for f_2 than what it does for f_1. This observation implies that the QoS performance guaranteed by a network service to an application is associated with application traffic characteristics as well as the QoS capability of the network.

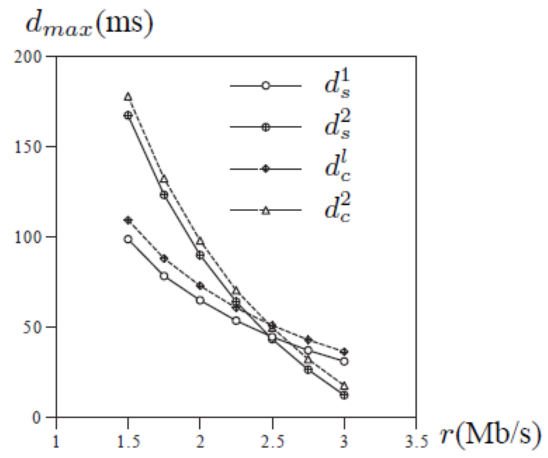

FIGURE 4: Delay performance prediction for a composite network service

We also predicted delay performance for A_1 and A_2 in a composite network service consisting of three network domains. To simplify the analysis, we assume that the three domains provide an identical QoS descriptor for each application. We calculated the maximum end-to-end packet delay for the traffic flows f_1 and f_2 that pass through the three domains, which are denoted by d_c^1 and d_c^2 respectively. We plotted the results of d_s^1 and d_s^2 with data for d_c^1 and d_c^2 in Figure 4. From this figure we can see that d_c^1 and d_c^2 are also decreasing function of available bandwidth \$r\$. This figure also shows that for any available bandwidth r, $d_s^1 < d_c^1$ and $d_s^2 < d_c^2$. This implies that the maximum packet delay guaranteed to each application by the composite network service is always greater than what is guaranteed by a single network service in the composite service. An interesting observation in Figure 4 is that for each application, the delay upper bound achieved by the composite network service is only slightly greater than the maximum delay guaranteed by a single service, which is much less than the summation of the maximum delays of all domains passed by the traffic flow.

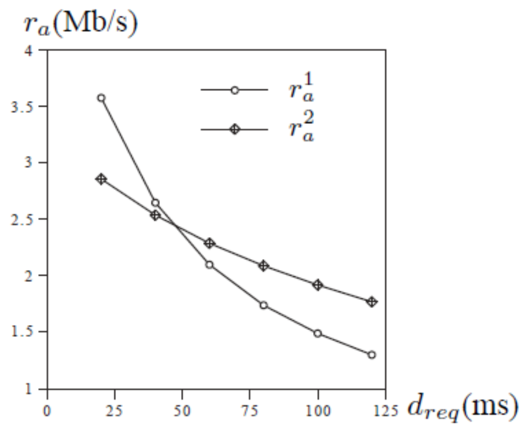

FIGURE 5: Bandwidth allocation for delay performance guarantee

We also analyzed the bandwidth allocation requirements in a network service to guarantee a set of delay upper-bounds for the two applications. The results are plotted in Figure 5, where r_a^1 and r_a^2 are respectively the required amounts of bandwidth for applications A_1 and A_2 to achieve the required delay upper-bound d_{req}. From this figure we can see that both r_a^1 and r_a^2 are decreasing functions of d_{req}. This means that the tighter the required delay upper-bound is, the more bandwidth must be allocated in the network to guarantee it. Figure 5 also shows that r_a^1 decreases with the increment of d_{req} faster than r_a^2 does. This implies that in the same network service, applications with different traffic load characteristics need different amounts bandwidth to achieve the same level of delay performance. This also justifies the necessary of having a traffic load descriptor *L* as part of the networking demand profile *P* for performance-based network service discovery.

6. NETWORK SERVICE DESCRIPTION AND DISCOVERY SYSTEM

In this section, we discuss a system for realizing the network service description and discovery technologies developed in previous sections. Figure 6 shows the structure of this system, which consists of a Network Service Broker (NSB), a Network Service Registry (NSR), multiple Network Service Providers (NSPs), and a Network Service Consumer (NSC). A network service provider could be single networking system, a network domain with multiple networking systems, or a collection of network domains that belong to an Internet Service Provider (ISP). The network service consumer could be user equipment, or a networking application, or maybe a network service provider that needs to access networking resources managed by other network service providers.

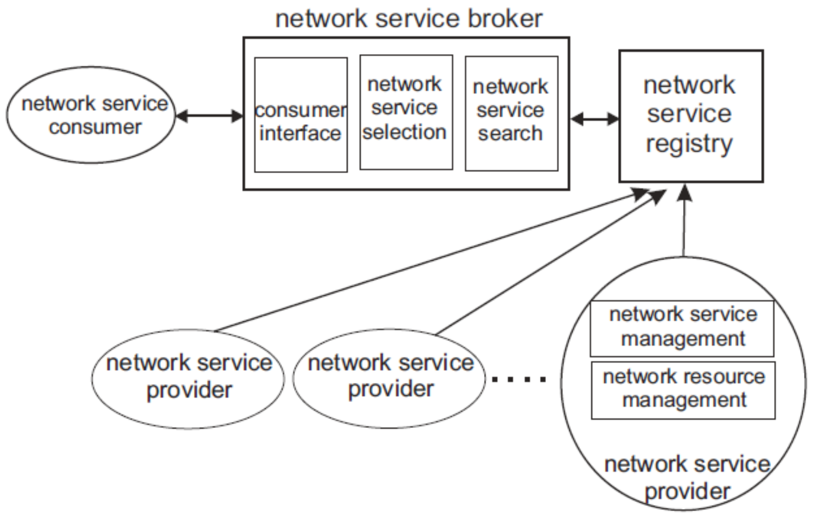

FIGURE 6: A system structure for network service description and discovery.

Each network service provider is required to support two functional modules for realizing the service-oriented networking paradigm: a Network Service Management (NSM) module and a Network Resource Management (NRM) module. The NSM module registers the network service at the broker, collects network state information from the NRM module to compile a network service description (including the capability matrix **C**), and publishes the description at the service registry. The NSM module is also responsible for updating the network service description at the service registry when network information changes.

The network service broker consists of three modules: Consumer Interface (CI), Network Service Discovery (NSD), and Network Service Selection (NSS). When a network service consumer needs to discover a service provider for accessing networking resources, it submits a networking request to the service broker through the CI interface. This request includes a networking demand profile ***P***[***d***, ***L***, ***a***]. On receiving the networking request and the demand profile, the discovery module (NSD) of the broker accesses the service registry to find all available network services that provide a network route between the source and destination specified in ***d*** of the demand profile. These network services are selected as candidate services and their capability matrixes will be further analyzed by the broker. For each candidate service, the selection module (NSS) uses the QoS descriptor from the matrix **C** and the load descriptor *L* in the demand profile *P* to evaluate the achievable QoS performance that the service can offer to the consumer. Then the NSS module compares the predicted performance with the requirements given in the set ***a*** of the profile *P* and selects the service that meets all performance requirements given by the consumer. If multiple candidate network services meet the requirements, selection among them is based on other criteria such as service cost or load balance. After selecting a network service, the network service broker will send a reply message through the CI interface to the service consumer to notify the network selection result. Then the consumer may contact the selected network service provider to negotiate a Service Level Agreement (SLA). The NRM module in the network service provider may need to allocate a certain amount of resources for QoS provisioning. After resource reservation, the NSM module of the service provider will accordingly update the capability matrix **C** and republish the latest service description at the service registry. The interactions among service consumer, broker, registry, and provider for network service description and discovery is shown in Figure 7.

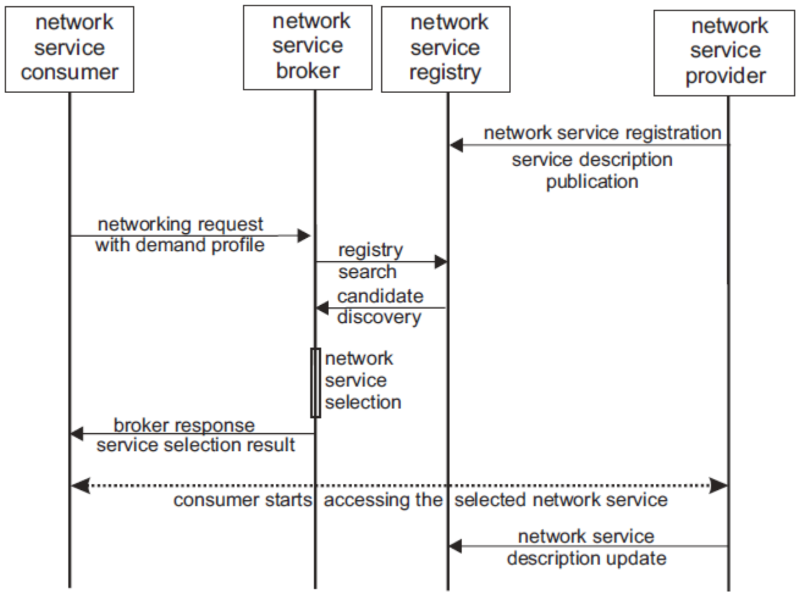
network service consumer
network service broker
network service registry
network service provider
network service registration
service description publication
networking request with demand profile
registry search
candidate discovery
network service selection
broker response service selection result
consumer starts accessing the selected network service
network service description update

FIGURE 7: Interaction procedure for network service description and discovery.

In this paper we focus on capability description and QoS-based selection for network services. Other technologies for implementing the network description and discovery system, including publishing service descriptions, maintaining service descriptions at the registry, and searching the registry for candidate networks can be implemented based the current SOA service description and discovery standards [32, 34, 35]. The negotiation and establishment of a service level agreement between the network service consumer and provider can be implemented based on the WS-Agreement specification [31].

In a large scale dynamic networking environment such as the Internet, the states and capability information of various networking systems change frequently. Therefore keeping the latest network description information at the network service registry is significant for discovering and selecting the appropriate network services. However, republishing the entire service description, including the capability matrix **C**, whenever network state changes cause a large amount of communications and management overhead between service providers and the registry, and make the service registry a performance bottleneck. In order to solve this problem, an event-driven subscription-notification mechanism can be applied to reduce the overhead caused by frequent description update; thus improving the overall performance of the service discovery system.

Event-driven processing and notification introduces a notification pattern for SOA implementations. In this pattern an information producer sends one-way notification messages to one or more interested receivers. The message typically carries information about an event that has occurred. The message receivers must register to the producer before receiving the notifications. The OASIS Web Service Notification [33] is a family of related specifications that define a standard approach to notification using a topic-based subscription-publication pattern. Following this pattern, the network service registry can subscribe to a network service provider and specify a set of network states as subscription topics. Then, the service registry will receive a notification message from the service provider whenever a network state subscribed as a topic changes. A threshold can also be set to each subscription topic so that update notification is only triggered when the amount of change in that topic is greater than the threshold. In this way, the network service registry can obtain the latest network state and capability information for supporting real-time network discovery and selection. Since an update only happens when a network state changes more than a pre-specified threshold and the notification message contains only the changed states instead of the entire description document, this updating mechanism can greatly reduce communications and management overhead.

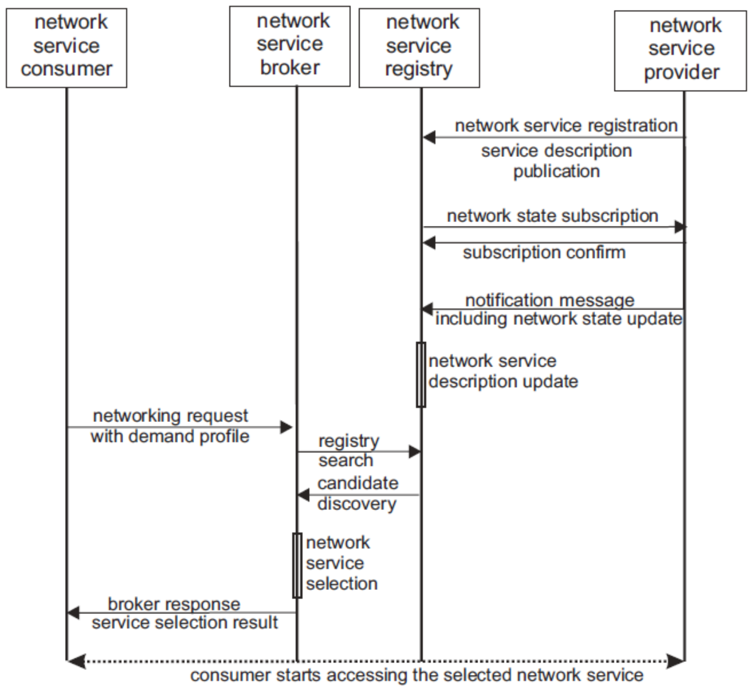
network service consumer
network service broker
network service registry
network service provider
network service registration
service description publication
network state subscription
subscription confirm
notification message including network state update
network service description update
networking request with demand profile
registry search
candidate discovery
network service selection
broker response service selection result
consumer starts accessing the selected network service

FIGURE 8: Event driven network service description update.

The interaction procedure for network service description and discovery with event-driven information update is shown in Figure 8. When a network service provider first time publishes its network service description at the service registry, the registry will also subscribe itself to the service provider to receive notification for network state change. The registry can specify what network states are subscription topics. After this registration-subscription procedure completed, the network service registry will be notified whenever the specified subscription topics change in the network service. Then the description for this network service will be updated at the registry accordingly.

The performance of the network service description and discovery system could be further improved by reducing the information updating overhead through a partial publication technology. This technology allows network service providers publish only part of their service descriptions that are relatively static at the service registry. This part of description could include information such as the network types (e.g., connectionless IP network or circuit switching telecommunication network, cellular mobile network or WiFi WLAN), network service operators (e.g., AT&T or Verizon), and also the connectivity parameters of the capability matrix **C**, which are relatively stable for typical networks. If a network service is selected as a candidate, then the service selection module of the broker will contact the NSM module in the provider of this service to retrieve additional information needed for decision making on network service selection, for example retrieving the QoS descriptor of the matrix **C** for performance evaluation. In this way the dynamic network state information such as the QoS capability descriptors of each network service is maintained within the service provider itself instead of at the service registry; thus can be updated in real time without generating communication overhead between networks and the service registry. The partial publication technology can be used together with the event-driven update mechanism. Update on the published part of description can be implemented by the notifications from the network service provider. Since only the relatively static part of description is published at the registry, the update frequency and the messaging load will be further reduced. Figure 9 shows the interactions for network service discovery and selection with partial description publication.

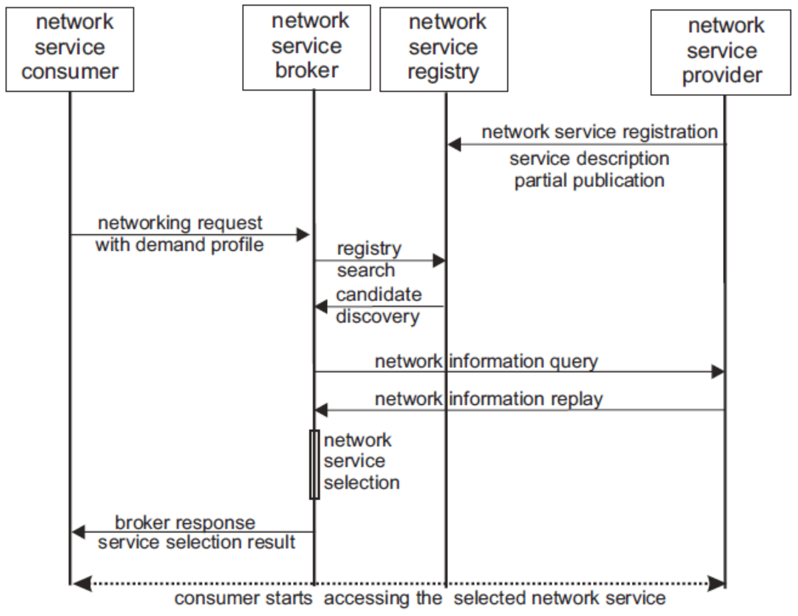
network service consumer
network service broker
network service registry
network service provider
network service registration
service description partial publication
networking request with demand profile
registry search
candidate discovery
network information query
network information replay
network service selection
broker response
service selection result
consumer starts accessing the selected network service

FIGURE 9: Partial publication of network service description.

7. CONCLUSIONS

The next generation Internet will be highly diversified in both underlying networking technologies and supported networking applications. Therefore coordination among heterogeneous networks to support the wide spectrum of application requirements is significant for building the next generation Internet. Although recently various research projects started addressing this challenge from different aspects, the notion of network virtualization plays a crucial role in all the efforts. The Service-Oriented Architecture, which has been widely applied in Web services and Grid/Cloud computing areas, provides a promising approach to supporting network virtualization. Network service capability description and performance-based network service selection are keys to enabling the application of SOA in the next generation Internet.

In this paper we developed a general approach to describing service capabilities of heterogeneous networking systems and a technology for selecting the network services that can guarantee application performance requirements. The core component of the network description approach is a capability matrix that describes both connectivity and QoS capability of a network service. The key of the network selection technology is to predict the achievable performance for a network service and check it against the performance requirement specified by the application. In order to allow various applications to specifying their networking requests, this paper gives a general demand profile that includes a traffic load descriptor and a performance requirement set. We also developed a scheme for allocation resources in network services for QoS provisioning. A system structure for realizing the network service description and discovery technologies is also described in this paper. In a large scale dynamic networking environment such as the Internet, keeping network service descriptions up-to-date is challenging and may cause significant communication and management load. We proposed an event-driven information update mechanism and a partial description publishing technology that may significantly improve the performance of the network service description and discovery system. Our analysis methods and the developed technologies for network service description, discovery, resource allocation, and information update are general and independent of network implementations; thus are applicable to heterogeneous networking systems in the next generation Internet. The service description and performance analysis techniques are also flexible to support composite network services that comprise multiple networking systems.

8. REFERENCES

1. R. J. Al-Ali, O. M. Rana and D. W. Walker. “G-QoSM: Grid service discovery using QoS properties". J. of Computing and Informatics, 21(4):1--15, 2002

2. R. J. Al-Ali, A. ShaikhAli, O. M. Rana and D. W. Walker. “Supporting QoS-based discovery in service-oriented grids". In Proceedings of the 17th Intl. Parallel and Distributed Processing symposium, 2003

3. E. Al-Marsri and Q. H. Mahmoud. “QoS-based discovery and ranking of Web services". In Proceedings of the 16th IEEE International Conference on Computer Communications and Networks, August 2007

4. E. Ambrosi, M. Bianchi, C. Gaibisso, G. Gambosi and F. Lombardi. “A system for predicting the run-time behavior of Web Services". In Proceedings of the 2005 International Conference on Services Systems and Services Management, June 2005

5. T. Anderson, L. Peterson, S. Shenker and J. Turner. “Overcoming the Internet impasses through virtualization". IEEE Computer Magazine, 38(4):34--41, 2005

6. F. Belqasmi, R. Glitho and R. Dssouli. “Ambient Network Composition". IEEE Network Magazine, 22(4):6--12, 2008

7. J. L. Boudec and P. Thiran. “Network calculus: a theory of deterministic queueing systems for the Internet", Springer Verlag LNCS 2050, June 2001

8. M. Butto, E. Cavallero and A. Tonietti. “Effectiveness of the leaky bucket policing mechanisms in ATM networks". IEEE J. Select. Areas Commun., 9(4):335--342, 1991

9. K. Channabasavaiah, K. Holley and E. Tuggle. “Migrating to a Service-Oriented Archiecture". IBM DeveloperWorks, Dec. 2003

10. N. Feamster, L. Gao and J. Rexford. “How to lease the Internet in your spare time". ACM SIGCOMM Computer Communications Review, 37(1):61--64, 2007

11. P. Fizek and M. Reisslein. “MPEG-4 and H.263 video traces for network performance evaluation". IEEE Network Magazine, 15(6):40--54, 2001

12. R. Douville, J.-L. Le Roux, J.-L. Rougier and S. Secci. “A service plane over the PCE architecture for automatic multidomain connection-oriented services". IEEE Communications Magazine, 46(6):90--102, 2008.

13. T. S. Eugene Ng and H. Zhang. “Predicting Internet network distance with coordinates-based approaches". In Proceedings of IEEE INFOCOM'02, New York, June 2002

14. J. Ham, P. Grosso, R. Pol, A. Toonk and C. de Laat. “Using the network description lanaguage in optical networks''. In Proceedings of the 10th IFIP/IEEE Intl. Symposium on Integrated Network Management, May 2007

15. A. Kind, X. Dimitropoulos, S. Denazis and B. Claise. “Advanced network monitoring brings life to the wareness plane''. IEEE Communicatins Magazine, 46(10):140--146, 2008

16. S. Lacour, C. Perez and T. PriolA. “Network topology description model for {G}rid application deployment''. In Proceedings of the 5th IEEE/ACM Intl. Workshop on Grid Computing, Nov. 2004

17. T. Lehman, J. Sobieski and B. Jabbari. “DRAGON: A Framework for Service Provisioning in Heterogeneous Grid Networks''. IEEE Communications Magazine, 44(3):84--90, 2006

18. M. Li, B. Yu, O. Rana and Z. Wang. “Grid service discovery with rough sets''. IEEE Transcations on Knowledge and Data Engineering, 20(6):851--862, 2008

19. A. K. Parekh and R. G. Gallager. “A generalized processor sharing approach to flow control in integrated services networks: the single-node case''. IEEE/ACM Trans. Networking, 1(3):344--357, 1993

20. R. Prasad, M. Murray, C. Dovrolis and K. Claffy. “Bandwidth estimation: metrics, measurement techniques, and tools''. IEEE Network Magazine, 17(6):27--35, 2003

21. G. Shao and F. Berman. “Using effective network views to promote distributed application performance''. In Proceedings of the 1999 Intl. Conf. on Parallel and Distributed Processing Techniques and Applications, June 1999

22. D. Stiliadis and A. Varma. “Latency-rate servers: a general model for analysis of traffic scheduling algorithms''. IEEE/ACM Trans. Networking, 6(5):611--624, 1998

23. J. Turner and D. E. Taylor. “Diversifying the Internet''. In Proceedings of IEEE Globecom 2005, Nov. 2005

24. C. Wan, C. Ullrich, L. Chen, R. Huang, J. Luo and A. Shi. “On solving QoS-aware service selection problem with service composition''. In Proceedings of the 7th Internatinal Conference on Grid and Cooperative Computing, Oct. 2008

25. S. Weerawarana, F. Curbera, F. Leymann, T. Storey and D. F. Ferguson. “Web Services Platform Architecture''. Prentice Hall, 2005.

26. T. Yu and K-J. Lin. “The design of QoS broker algorithms for QoS-capable Web Services''. International Journal of Web Services Research, 1(4):10--16, 2004

27. T. Yu and K-J. Lin. “Service selection algorithm for Web services with end-to-end QoS constraints''. Journal of Information Systems and E-Business Management, 3(2):20--30, 2005

28. ITU-T, “Functional Requirements and Architecture of the NGN Release 1,'' Recommendation Y.2012, Sept. 2006

29. Open Mobile Alliance, “OMA Enabler Releases and Specifications - OMA Service Environment Architecture Document,'' Nov. 2007

30. Open Grid Forum (OGF) OGSA-WG, “The Open Grid Service Architecture, version 1.0,'' Jan. 2005

31. Open Grid Forum (OGF) “Web Services Agreement Specification (WS-Agreement)," March 1007, available online: http://www.ogf.org/documents/GFD.107.pdf

32. Organization for the Advancement of Structured Information Standards (OASIS), “Universal Description, Discovery and Integration Version 3.0.2," Feb. 2005, available online: http://www.oasis-open.org/specs/

33. Organization for the Advancement of Structured Information Standards (OASIS), “Web Services Base Notification (WS-BaseNotification) v1.3," Oct. 2006

34. Organization for the Advancement of Structured Information Standards (OASIS), “Web Services Resource Framework (WSRF) v1.2," April 2006

35. World Wide Web Consortium (W3C), “Web Service Description Language Version 2," March 2006

36. World Wide Web Consortium (W3C), “Simple Object Access Protocol (SOAP) Version 1.2," April 2007

37. World Wide Web Consortium (W3C), “Web Services Policy Framwork (WS-Policy) Version 1.5," Sept. 2007, available online: http://www.w3.org/TR/ws-policy/

38. World Wide Web Consortium (W3C), “Web Services Policy Attachment (WS-PolicyAttachment) Version 1.5," Sept. 2007, available online: http://www.w3.org/TR/ws-policy-attach/

A Cross-Layer Packet Loss Identification Scheme to Improve TCP Veno Performance

Sachin Shetty sshetty@tnstate.edu
Department of Electrical and Computer Engineering
Tennessee State University
3500 John A. Merritt Blvd
Nashville, TN 37209

Ying Tang tang@rowan.edu
Department of Electrical and Computer Engineering
Rowan University
201 Mullica Hill Rd
Glassboro, NJ 08028

William Collani wcollani@gmail.com
Department of Electrical and Computer Engineering
Rowan University
201 Mullica Hill Rd
Glassboro, NJ 08028

Abstract

In wired-cum-wireless networks, one of the main design challenges for TCP is to accurately distinguish congestion losses from random losses caused due to channel noise and interference. TCP Veno has been widely deployed in wireless networks to address this challenge. TCP Veno has been demonstrated to show better performance than TCP Reno in wired-cum-wireless environments. However, TCP Veno does not take into consideration the effects of channel noise and interference on packet losses, which impacts the accuracy of TCP Veno's packet loss identification and thereby causes underutilization of bandwidth in lightly-loaded networks. In this paper, we propose TCP Venoplus which contains two refinements to improve the performance of TCP Veno in lightly-loaded networks. The first refinement incorporates a new variable, congestion loss window, into TCP Veno. This new variable will aid TCP Veno in proper utilization of bandwidth in presence of lightly-loaded traffic. The second refinement involves procuring power level of received packet from the MAC layer to aid in better detection of random losses. The simulation results demonstrate that, the two refinements in TCP Venoplus can significantly improve Veno's throughput with accurate packet loss identification and without compromising fairness.

Keywords – Transmission Control Protocol (TCP), System-on-Chip (SoC), congestion control, packet loss differentiation

1. INTRODUCTION

In recent times, communication networks have evolved greatly. Packet switching technologies have successfully merged the traditional voice networks and data networks into a converged and integrated multimedia network. This converged integrated network has been extended to incorporate varied flavors of wireless technologies. Transmission control protocol (TCP) has become the dominant communication protocol suite in almost all the Internet application for reliable data transfer services [1]. The great success of TCP is due to its end-to-end congestion control scheme, which is designed for fairness and friendliness with respect to coexistent TCP flows.

With the proliferation of mobile computing in recent years, more and more devices are communicating through wireless links. These links are often characterized by high random bit error rates (BER) due to channel fading or noise, and intermittent connectivity due to handoffs. Therefore, network protocols, originally designed for wired networks, have to be adapted for such lossy environment. Nowadays, about 90% of the Internet traffics are carried by TCP. TCP needs to depart from its original wired network oriented design and evolve to meet the challenges introduced by the wireless portion of the network. Transmission Control Protocol (TCP) has become the dominant communication suite in today's networking applications. TCP was designed ideally for wired network with low BER, where congestion is the primary source of packet losses. Although TCP was initially designed and optimized for wired networks, the growing popularity of wireless data applications has lead wireless networks such as WLAN to extend TCP to wireless communications as well. The initial objective of TCP was to efficiently use the available bandwidth in the network and to avoid overloading the network (and the resulting packet losses) by appropriately throttling the senders' transmission rates. Network congestion is deemed to be the underlying reason for packet losses [3]. Consequently, TCP performance is often unsatisfactory when used in wireless networks and requires various improvement techniques [2], [8], [9], [10], [12]. When applied to the wireless environment in which packet losses are often induced by noise, link errors, and reasons other than network congestion (random errors), TCP congestion control and avoidance mechanism [6] becomes incapable of dealing with the mixed packet losses. In fact, when a random loss occurs, TCP actually backs down the sending rate to reduce, what it perceives as, congestion in the network [4], resulting in significant performance degradation. There are three key factors responsible for the unsatisfactory performance of TCP in wireless networks. The first factor is that the radio link quality in wireless networks can fluctuate greatly in time due to channel fading and user mobility, leading to a high packet loss which is non-congestion loss[4][5][6]. The second factor is variability of transmission time and delay [7]. Finally, the third factor is high delay variability caused by radio link quality in wireless networks. A form of high delay variability, referred to as delay spike, is a sudden, drastic increase in delay for a particular packet or a few consecutive packets, relative to the delay for the preceding and following packets.

TCP's congestion control scheme is based on the fundamental premise that packet loss is an indicator for network congestion. Therefore, TCP senders react to packet loss by reducing its sending throughput. However, this premise is inaccurate as increasing number of wireless links are integrated into the Internet. Packet losses in wireless links are overwhelmingly caused due to bit errors, instead of buffer overflow at routers [2], [14], [15], [16]. This cause of packet loss due to bit errors has been largely ignored by TCP without any pertinent refinement. The result of such failure is that TCP will "blindly" scale back its sending rate upon packet losses with identical penalty, regardless of the reason of loss occurrence. So, TCP will suffer unnecessary performance degradation.

In recent years, different schemes have been proposed to aid TCP in distinguishing congestion losses from bit errors. We present the details of these schemes in the next Section. To provide an end-to-end congestion control a sender-side variation of TCP Reno, TCP Veno [2] was developed. Veno purely uses the information available at the sender to make decisions regarding the types of losses. The detection of packet losses is obtained from estimated congestion information. A threshold value β is used to make the decision. If the estimated congestion level in the network exceeds the value of β, the packet loss is detected as a congestion loss; else it is classified as a loss due to random errors. In case of a random loss, the congestion window is reduced by a factor of 1/5, instead of halving the congestion window [2]. Therefore, if the estimated congestion information is accurate, the packet losses will be accurately detected at all times.

But TCP Veno has been demonstrated to not perform effectively in certain network scenarios. In [13], it has been shown that due to some uncertainties in network the actual network state could be associated with an irrelevant packet loss; or a packet loss is linked with the network state that is mistakenly estimated. These factors imply that the loss due to transient congestion is also a factor which can influence the prediction of packet losses. In [14], the authors claimed that the transient congestion period occurs so fast that a corresponding packet loss looks just like an episode of wireless loss. In presence of bursty traffic, it has been shown that Veno's packet loss identification scheme is not very accurate [15]. More recently, it has been demonstrated that Veno suffers from server bandwidth under utilization when operating under lightly loaded wireless networks [16].

In recent times, with the rapid progress of deep submicron technology, System-on-Chip (SoC) has revolutionized the design of Integrated Circuits, allowing all components of a computer or other electronic system into a single integrated circuit (i.e., chip). One of the applications in networking is offloading TCP/IP processing into an embedded device called TCP Offload Engine (TOE). TOE provides an emerging solution to release communication systems from burdened conventional TCP/IP stack, and significantly improves the network utilization [11]. Modern servers with multi-processor and multi-core architectures are now equipped with TOE to boost throughput and reduce CPU overload. The advent of SoC technology further opens a new venue for cross-layer design in wireless networking since the traditional layered architecture served well for wired networks is not suitable for wireless networks. More specifically, the integration of all layers of networking into a single chip presents the following two advantages: (1) the impact of breaking end-to-end network semantics is negligible; and (2) the implementation of cross-layer communication becomes easier.

Motivated by these observations, this paper addresses the aforementioned two problems together to improve TCP performance in both lightly and heavily loaded wireless networks. In this paper, we propose TCP Venoplus which incorporates two refinements in the TCP Veno scheme. The two refinements improve the congestion state measurement in Veno. By taking advantages of SoC technologies, Venoplus uses received signal strength information (RSSI) to compute BER and duplicate acknowledgements and timeouts to estimate congestion loss. The refinements in Venoplus are on the same line as the enhancedVeno scheme proposed in [16]. Enhanced Veno describes a scheme to distinguish congestion losses from random losses for lightly-loaded wireless networks. But enhancedVeno assumes that the nature of random errors in the link is known *a priori*. This is not practical in most wireless communication links due to the randomness associated with noise and interference sources. Venoplus differs from enhancedVeno by incorporating cross-layer functionality to compute random losses. The cross-layer property allows the TCP layer to learn RSSI for every received packet from the MAC layer.

Two new variables, congestion loss window and random loss rate are added to TCP Veno. The congestion loss window variable aids TCP Veno in improving bandwidth utilization without compromising fairness. The random loss rate variable incorporates the information procured from BER to signify the number of packet losses due to random errors. The computation of the random loss rate involves the implementation of cross-layer functionality. As stated earlier, the implementation complexity in terms of cross-layer communication becomes negligible with recent SoC advancements. Extensive simulations demonstrate that the throughput and accurate detection of congestion and random losses are much improved than TCP Veno. We also observed that Venoplus yields better fairness in presence of competing TCP connections.

The remainder of this paper is organized as follows: Section 2 presents the related work. In Section 3 we analyze the issue of bandwidth underutilization in TCP Veno and present the details for TCP Venoplus. The simulation and performance evaluation is described in Section 4. In Section 5, we present conclusions and future work.

2. RELATED WORK

In recent years, different schemes have been proposed to aid TCP in distinguishing congestion losses from random errors. More specifically, the TCP schemes address the following two problems (1) how to distinguish a packet loss due to congestion and the corruption of a packet due to random losses; and (2) how to refine the congestion-window adjustment parameters according to network conditions. Many research efforts have been directed towards the modification of TCP protocol to address these two problems. All these research efforts fall under two main categories. The two categories are: Wireless TCP Refinements and Routers with RED and ECN.

2.1 Wireless TCP Refinements

Existing TCP schemes suffer from severe performance degradation in hybrid wired-cum-wireless networks [18], [19]. There have been numerous research efforts to alleviate this problem. These research efforts can be categorized into the split mode approach, link layer approach, and end-to-end TCP modifications. In the split mode approach [20]–[22], the traffic in a wireless network is protected from the wired network by separating the TCP connections of the wired and wireless networks. The base station is responsible for the recovery of the packet losses caused by wireless links. This approach requires large buffers at base stations which violates the end-to-end TCP semantics. The link layer approach [23]–[26] rectifies wireless link errors by employing a suite of techniques such as forward error correction (FEC), automatic repeat request (ARQ), and explicit loss notification (ELN). However, these techniques require protocols at different layers to interact and coordinate closely, which increase the complexity of protocol implementation. In the end-to-end approach, TCP senders and receivers are responsible for the flow control. TCP-Peach [27] is particularly designed for the satellite communication environment. TCP-Peach replaces slow start and fast recovery phases with sudden start and rapid recovery, respectively. In sudden start and rapid recovery, the sender probes the available network bandwidth in only one RTT with the help of low-priority dummy packets. An important assumption made by TCP-Peach is that the routers must support priority queuing. In TCP-Peach Plus [28], the actual data packets with lower priority replace the low-priority dummy packets as the probing packets to further improve the throughput. TCP-Westwood [29] is a rate based end-to-end approach, in which the sender estimates the available network bandwidth dynamically by measuring and averaging the rate of returning ACKs. TCP-Westwood claims improved performance over TCP-Reno and -Sack, while achieving fairness and friendliness. The end-to-end approach maintains the network layer structure and requires minimum modification at end hosts and router.

2.2 RED and ECN enabled routers

In this approach, routers play a significant role in controlling TCP transmission rates by implementing active queue management (AQM). Random early detection (RED), proposed in [30], is an AQM scheme implemented in routers that informs the sender of incipient congestion by probabilistically dropping packets before the buffer overflows. The packet drop triggers the receiver to send DUPACKs to the sender. The sender is therefore able to adjust its window size in response to the packet drop by means of congestion control. The early packet drop prevents the router to enter the fully congested state. By doing so, the average queue length at the router can be kept small, hence reducing the queueing delay and improving the TCP throughput. Explicit congestion notification (ECN) [31] is an extension to RED. Instead of randomly early dropping packets, an ECN router marks packets to alert the sender of incipient congestion. ECN is an explicit signaling mechanism designed to convey network congestion information from routers to end stations; however, since the signaling only uses one bit for such congestion information, the information conveyed is not quantitative. For TCP flow control, ECN works by configuring the intermediate router to mark packets with congestion experienced (CE) bit in the IP header when the router's average queue occupancy exceeds a threshold, so that the TCP receiver can echo this information back to the sender via ACK by setting the explicit congestion echo (ECE) bit in the TCP header. TCP Jersey [13] integrates ECN with a modified version of TCP Westwood. Although its throughput performance is improved, the additional requirement on network routers to provide information about the onset of congestion limits their practical applications.

3. TCP VenoPlus

In this paper, we propose TCP VenoPlus which incorporates a congestion loss window to keep track of the level of congestion in the bottleneck links and leverages the availability of signal strength information for the received packet at the MAC layer to accurately detect the presence of random losses due to noise and/or interference on the physical links. VenoPlus adjusts the congestion window in a smart fashion by utilizing the information obtained from the congestion loss window and the signal strength of the received packet. The congestion loss window is calculated as follows: Considering a sequence of packet losses during the lifetime of a TCP connection. Let W_i be the packet loss monitoring window. The monitoring window represents the number of packets lost to be observed in the current window. The window is a configurable parameter which represents the number of received packets for observation of packet losses. Let T_i be the time stamp for window W_i. The congestion losses are computed during every window cycle. The number of congestion losses occurring during the current window W_i is called the congestion loss (C_i). The congestion loss window $conw_i$ can be calculated as follows:

$$conw_i = C_i \,/\, (T_i - T_{i-1}) \qquad (2)$$

Next, we will show how TCP Venoplus utilizes the received signal strength information (*RSSI*) for a packet to calculate random losses due to noise and interference. Each radio receiver in a wireless node is equipped with power sensitivity, which allows the receiver to detect and decode signals with strength larger than this sensitivity. There are two threshold values when receiving radio signals: receive threshold (*RXThresh*) and carrier sense threshold (*CSThresh*). If the power of the received signal is higher than *RXThresh*, it is regarded as a valid packet and passed to the TCP layer. During the current monitoring window cycle, Venoplus obtains the *RSSI* for every received packet from the MAC layer. This requires a cross-layer implementation to facilitate transfer of *RSSI* values between the MAC layer and TCP layer. In recent years, cross-layer approaches have gained significant attention, due to their ability to allow critical information of the wireless medium in the MAC and physical layers to be shared with higher layers, in order to provide efficient methods of allocating network resources and applications over the Internet [19].
The packets whose RSSI is less than the *RXThresh* or *CSThresh* are dropped by the MAC layer. But Venoplus procures the number of dropped packets (R_i) from the MAC layer and computes the random loss rate ($ranl_i$). R_i is calculated at the MAC layer by keeping track of the received signal strength (RSS) of each packet. If the RSS value falls below a certain threshold the packet is dropped.

$$ranl_i = R_i \,/ (T_i - T_{i-1}) \qquad (3)$$

We use a low-pass filter to calculate $conw_i$ and $ranl_i$. For the low pass filter we use the exponential weighted moving average (EWMA) [18].

```
**if** (congestion loss) **then**
    $conw_{inst} = \alpha conw + (1-\alpha) conw$
**else** if (random loss) **then**
    $ranl_{inst} = \alpha ranl + (1-\alpha) ranl$
**end if**
```

Figure 1 illustrates the process of computation of congestion loss window and random loss rates. The packet train is categorized into different windows during a given period of time. The values for *conw* and *ranl* are calculated during every window. Having calculated the values for *conw* and *ranl*, we now present the packet loss identification scheme:

```
if ($conw_i > conw_{i-1}$ and $ranl_i > ranl_{i-1}$) then
   cwnd=cwnd x 1/2; { Packet losses due to congestion and random errors}
if ($conw_i < conw_{i-1}$ and $ranl_i > ranl_{i-1}$) then
   cwnd=cwnd x 4/5; { Packet losses due to random errors}
if ($conw_i > conw_{i-1}$ and $ranl_i < ranl_{i-1}$) then
   cwnd=cwnd x 1/2; { Packet losses due to congestion}
if ($conw_i < conw_{i-1}$ and $ranl_i < ranl_{i-1}$) then
   cwnd=cwnd ; { No packet losses}
```

The packet identification scheme illustrates four different scenarios which are summarized as follows: 1) If $conw_i > conw_{i-1}$ and $ranl_i > ranl_{i-1}$, the network suffers from losses due to congestion and random errors in the link simultaneously. This situation requires slowing down the transmission of packets which is reflected by the reduction of the congestion window by half. 2) If $conw_i < conw_{i-1}$ and $ranl_i > ranl_{i-1}$, the packet losses can be attributed to random errors in the link only. The congestion window is cut down by 1/5. 3) If $conw_i > conw_{i-1}$ and $ranl_i < ranl_{i-1}$, the packet losses can be attributed to congestion losses in the network. The congestion window is now cut down by 1/2. 4) Finally, if $conw_i < conw_{i-1}$ and $ranl_i < ranl_{i-1}$, the congestion in the network and the random errors in the link are not getting worse, which means that the congestion window remains unchanged.

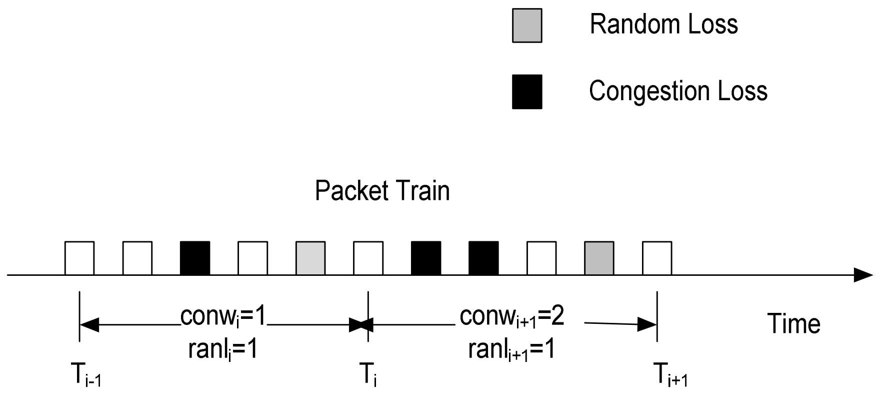

Figure 1: Computation of congestion loss window and random loss rate.

4. SIMULATION & EVALUATION

In this section, we will evaluate the following performance metrics of VenoPlus as compared to Veno: throughput, and fairness. The comparison will be performed with varied number of nodes and random loss rates.

4.1 Simulation Model

We use network simulator ns-2 to conduct the simulations. Figure 2 illustrates the heterogeneous wired-cum-wireless topology. The wired nodes represent the source of traffic and the wireless nodes represent the sinks. The wired links between R2 and the traffic sources are set with 10 Mbps bandwidth, 0.1 ms roundtrip time and buffer size of 50 packets. The wireless links between R1 and the traffic sink are set with 10Mbps bandwidth, 0.1 ms round-trip time, and buffer size of

50 packets. The link between R1 and R2 represents the wired bottleneck link. The bandwidth of the bottleneck link is set to be 10 Mbps and the propagation delay is 80 ms. The random loss rate in the wireless links changes from 0.0001 to 0.1 in packet unit. The shadowing model for physical links is appropriately configured to generate the desired random loss rates.

4.2 Throughput

To compare the throughput between Venoplus and Veno, multiple TCP connections ranging from 1 to 64 are setup to create a congestive bottleneck link. The throughput comparison is computed by a normalized throughput ratio. The normalized throughput is given by

$$TH_{norm} = \frac{TH_{venoplus}}{TH_{veno}}$$

where $TH_{venoplus}$ and TH_{veno} are the average throughput of flows for Venoplus and Veno respectively. The normalized throughput will signify the improvement in throughput obtained by Venoplus as compared to Veno.

Figure 3 illustrates the normalized throughput for different number of TCP connections and bit error rates. The bit error rates range from 10^{-4} to 10^{-1}. As shown in Figure 3, TCP Venoplus is capable of better throughput improvements than Veno when the number of TCP connections is small. The maximum improvement is close to 80 % when the bit error rate is 10^{-3}. But as the number of TCP connections increase, which is an indicator of heavy network traffic, the throughput offered by Venoplus is closer to Veno. This proves our hypothesis that Venoplus is better than Veno when the network load is light and similar to Veno when the network load is heavy.

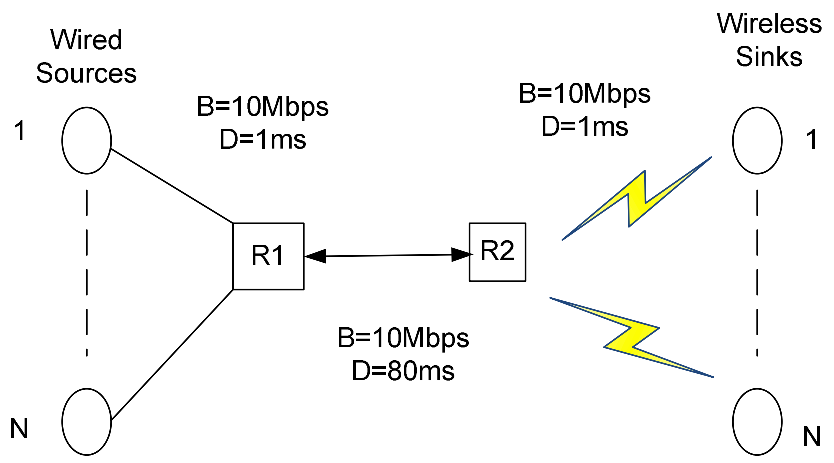

Fig. 2: Simulation Topology

4.3 Fairness

The goal of TCP Fairness is that if N TCP connections share same bottleneck link, each connection should get 1/N of link capacity. In our simulations, we set the value of N to be 64 and verify if each connection receives 1/64 of the link capacity. The fairness ratio is computed using the Jain Fairness index *f* which is defined as:

$$f = \frac{\left(\sum_{i=1}^{n} th_i\right)^2}{n\sum_{i=1}^{n} th_i^2}$$

where, n is the number of TCP connections, th_i is the throughput for the *i*th connection. The goal of TCP fairness is then to ensure that the value of f is closer to 1. Figure 4 illustrates the fairness ratio for Venoplus. The fairness ratio is close to 1 from light traffic to heavy traffic and for low bit error rate to high bit error rate. This proves that Venoplus is capable of providing better throughput without compromising fairness.

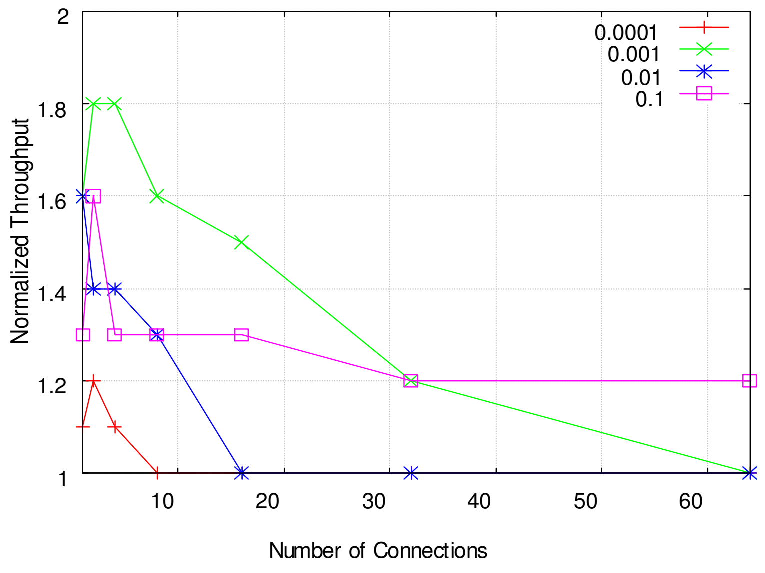

Figure 3: Throughput improvement of Venoplus in presence of different bit error rates and TCP connections.

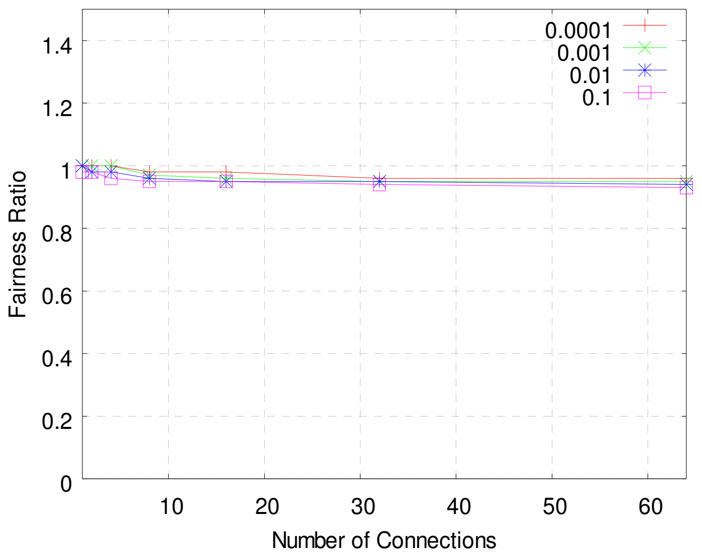

Figure 4: Fairness index of Venoplus

5. CONCLUSION & FUTURE WORK

TCP schemes have to adapt to the changing trends in modern networks. In this paper we address the problems with existing TCP schemes to accurately detect the reason for packet losses. We highlight the performance issues for TCP Veno in presence of bursty congestion and lightly loaded wireless networks. We propose TCP Venoplus which incorporates two refinements to TCP Veno to alleviate the performance degradation in lightly loaded wireless networks. Simulations results demonstrate that Venoplus can improve the accuracy of congestion loss identification in Veno in presence of light traffic. Venoplus provides significant improvement in throughput over Veno without sacrificing fairness.

For future work, we plan to extend the simulations to include bursty traffic and evaluate the performance of Venoplus in presence of transient congestion. We also plan to incorporate Venoplus in the TCP/IP suite of a Xilinx Virtex-II Pro FPGA and perform experimental evaluations.

6. ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation under award number DUE-0633512. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

7. REFERENCES

1. W. Richard Stevens, “TCP/IP Illustrated,” Addison-Wesley 1994.
2. C. P. Fu and S. C. Liew. “TCP Veno: TCP Enhancement for Transmission Over wireless Access Networks,” IEEE Journal on Selected Areas in Communication, February 2003.
3. V. Jacobson, "Congestion avoidance and control," SIGCOMM 88, ACM, Aug. 1988
4. C. L. Zhang, C. P. Fu, M. T. Yap, C. H. Foh, K. K. Wong, C. T. Lau, and E. M. K. Lai, “Dynamics comparison of TCP Reno and Veno,” in Proc. IEEE Globecom 2004.
5. C. P. Fu, W. Lu, and B. S. Lee, “TCP Veno revisited,” in Proc. IEEE Globecom 2003.
6. Y. Tian, K. Xu, and N. Ansari. “TCP in Wireless Environments: Problems and Solutions,” IEEE Radio Communications Magazine, March 2005.
7. A. Gurtov, “Effect of Delays on TCP Performance,” Proc of IFIP Personal Wireless Commun., Aug. 2001.
8. Balakrishnan, H., Padmanabhan, V. N., Seshan, S., and Katz, R. H., “A Comparison of Mechanisms for Improving TCP Performance over Lossy Links,” *IEEE/ACM Transactions on Networking*, Dec 1997
9. Balan, R.K.; Lee, B.P.; Kumar, K.R.R.; Jacob, L.; Seah, W. K. G., Ananda, A.L., “TCP HACK: TCP header checksum option to improve performance over lossy links,” *Proceedings of Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 309-318, 2001.
10. Comer, Douglas E. (2006). *Internetworking with TCP/IP* (5E ed.). Prentice Hall: Upper Saddle River, NJ.
11. Chung, S. M., Li, C. Y., Lee, H. H., Li, L. H., Tsai, Y. C. and Chen, C. C., “Design and implementation of the high speed TCP/IP Offload Engine,” *Proceedings of IEEE International Symposium on Communications and Information Technologies,* Oct. 17-19, 2007, pp. 574-579.
12. Krishnan, R., Allman, R. M., Partridge, C. and Sterbenz, J. P. G., “Explicit Transport Error Notification for Error-Prone Wireless and Satellite Networks,” *BBN Technical Report* No. 8333, BBN Technologies, Feb. 2002.
13. Xu, K., Tian, Y. and Ansari, N., “TCP-Jersey for Wireless IP Communications,” *IEEE Journal on Selected Areas in Communications*, Vol. 22, No. 4, 2004, pp. 747-756.

14. L. Brakmo, S. O'Malley, and L. Peterson. "TCP Vegas: new techniques for congestion detection and avoidance," SIGCOMM, ACM, 1994.
15. Z. Zou, C. Fu, B. Lee, "A refinement to improve TCP Veno performance under bursty congestion," in Proc. IEEE Globecom 2005.
16. B. Zhou, C. Fu, K. Zhang, C. Lau, C. Foh, "An enhancement of TCP Veno over light-load wireless networks," IEEE Communication Letters, 2006.
17. S. Shakkottai, T. S. Rappaport and P. C. Karlsson ,"Cross-layer Design for Wireless Networks,", IEEE Communications magazine, October, 2003.
18. T. V. Lakshman and U. Madhow, "The performance of TCP/IP for networks with high bandwidth-delay products and random loss," *IEEE/ACM Trans. Networking*, vol. 5, pp. 336–350, June 1997.
19. F. Lefevre and G. Vivier, "Understanding TCP's behavior over wireless links," in *Proc. Communications Vehicular Technology*, 2000, SCVT-2000, pp. 123–130.
20. V. Tsaoussidis and I. Matta, "Open issues on TCP for mobile computing," *J. Wireless Commun. Mobile Computi.*, vol. 2, no. 1, pp. 3–20, Feb. 2002.
21. A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for mobile hosts," in *Proc. ICDCS 95*, May 1995, pp. 136–143.
22. K. Wang and S. K. Tripathi, "Mobile-end transport protocol: An alternative to TCP/IP over wireless links," in *IEEE INFOCOM*, vol. 3, Mar. 1998, pp. 1046–1053.
23. H. Balakrishnan, S. Seshan, and R. H. Katz, "Improving reliable transport and handoff performance in cellular wireless networks," *ACM/Baltzer Wireless Networks J.*, vol. 1, no. 4, pp. 469–481, Dec. 1995.
24. S. Keshav and S. Morgan, "SMART retransmission: Performance with overload and random losses," in Proc. IEEE INFOCOM'97, vol. 3, 1997, pp. 1131–1138.
25. K. Ratnam and I. Matta, "WTCP: An efficient mechanism for improving TCP performance over wireless links," in Proc. Int Symp. Computers Communications, 1998, pp. 74–78.
26. H. Balakrishnan and R. H. Katz, "Explicit loss notification and wireless web performance," in Proc. IEEE GLOBECOM Internet Mini-Conf., Sydney, Australia, Nov. 1998.
27. I. F. Akyildiz, G. Morabito, and S. Palazzo, "TCP-Peach: A new congestion control scheme for satellite IP networks," IEEE/ACM Trans. Networking, vol. 9, pp. 307–321, June 2001
28. I. F. Akyildiz, X. Zhang, and J. Fang, "TCP-Peach+: Enhancement of TCP-Peach for satellite IP networks," *IEEE Commun. Lett.*, vol. 6, pp. 303–305, July 2002.
29. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," *ACM Mobicom*, pp. 287–297, July 2001.
30. S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Networking*, vol. 1, pp. 397–413, Aug. 1993.
31. S. Floyd, "TCP and explicit congestion notification," *ACM Comput.Commun. Rev.*, vol. 24, pp. 10–23, Oct. 1994.