# UC Davis

## UC Davis Previously Published Works


### Title

Fair queueing with service envelopes (FQSE): A cousin-fair hierarchical scheduler for subscriber access networks


### Permalink

https://escholarship.org/uc/item/6vt1p49m



### Journal

IEEE Journal on Selected Areas in Communications, 22(8)

### ISSN

0733-8716



### Authors

Kramer, Glen
Banerjee, A
Singhal, N K
et al.



### Publication Date

2004-10-01

Peer reviewed



eScholarship.org Powered by the California Digital Library
University of California

# Fair Queueing With Service Envelopes (FQSE): A Cousin-Fair Hierarchical Scheduler for Subscriber Access Networks

Glen Kramer, *Member, IEEE*, Amitabha Banerjee, *Student Member, IEEE*, Narendra K. Singhal, *Student Member, IEEE*, Biswanath Mukherjee, *Member, IEEE*, Sudhir Dixit, *Senior Member, IEEE*, and Yinghua Ye, *Member, IEEE*

***Abstract*—In this paper, we propose and investigate the characteristics of a fair queueing with service envelopes (FQSE) algorithm—a hierarchical fair-share scheduling algorithm for access networks based on a remote scheduling system such as Ethernet passive optical networks (EPON) or cable TV network. FQSE is designed to overcome the limiting factors of a typical remote scheduling system such as large control-plane delay, limited control-plane bandwidth, and significant queue switch-over overhead. The algorithm is based on a concept of *service envelope*—a function representing the fair allocation of resources based on a global network condition called satisfiability parameter (SP). We define properties of *cousin-fairness* and *sibling-fairness* and show the FQSE to be cousin-fair. FQSE is unique in that it is the only hierarchical algorithm that is simultaneously *cousin-fair*. Furthermore, we show the necessary techniques to adapt FQSE to variable-sized packet-based networks. We analyze FQSE performance in EPON serving 1024 independent queues and demonstrate FQSE's ability to provide guaranteed bandwidth to each queue and to share the excess bandwidth fairly.**

***Index Terms*—Ethernet passive optical networks (EPON), fair queueing, remote scheduling, subscriber access networks.**

## I. Introduction

OVER the past twenty years, a lot of attention has been given to the problem of fair scheduling and fair resource allocation. The authors' desire to revisit the scheduling problem came from their participation in IEEE 802.3ah "Ethernet in the First Mile" task force, a standard body chartered with extending the Ethernet standard (IEEE 802.3) into the access network area. Among other architectures, the task force is standardizing Ethernet passive optical networks (EPONs)—an architecture combining Ethernet operation (media-access control and variable-length packet format) with an all-fiber tree-based topology. We found, to our surprise, that among the multitude of existing fair-scheduling algorithms, no one is suitable for EPON or for a remote scheduling system, in general. We define a *remote scheduling system* as a scheduling (resource-sharing) domain in which the queues (customers) and the scheduler (server) are located at a large distance from one another. EPON is just one example of a remote scheduling system; other examples include wireless (cellular) or cable TV networks. The properties of a typical remote scheduling system such as large control-plane delay, limited control-plane bandwidth, and significant queue switch-over overhead do not allow easy adaptation of existing scheduling algorithms.

In this paper, we propose and investigate the characteristics of a new algorithm, called fair queueing with service envelopes (FQSE), which addresses the issues specific to remote-scheduling systems. FQSE is not only applicable to EPON systems, but can be generalized to other point-to-multipoint topologies, e.g., wireless networks or coax-tree networks.

### A. EPON—An Example of a Remote Scheduling System

An EPON is a point-to-multipoint (PtMP) optical network with no active elements in the signals' path from source to destination. The only interior elements used in EPON are passive optical components, such as optical fiber, splices, and splitters. EPON architecture saves cost by minimizing the number of optical transceivers, central office terminations, and fiber deployment. We refer the reader to [1] for an in-depth description of EPON architecture.

All transmissions in an EPON are performed between a head-end called optical line terminal (OLT) and tail-ends called optical network units (ONUs) (Fig. 1). The ONU serves either a single subscriber (fiber-to-the-home) or multiple subscribers (fiber-to-the-curb or fiber-to-the-multidwelling-unit). In the downstream direction, EPON is a broadcasting media; Ethernet packets transmitted by the OLT pass through a $1:N$ passive splitter or a cascade of splitters and reach each ONU. An ONU filters packets destined to its users and discards the rest (Fig. 1).

In the upstream direction (from the ONUs to the OLT), the ONUs need to employ some arbitration mechanism to avoid data collisions and fairly share the channel capacity. This is achieved by the OLT allocating (either statically or dynamically) a nonoverlapping transmission window (time slot) to each ONU. To enable time slot assignment, the IEEE 802.3ah task force is developing a multipoint control protocol (MPCP). MPCP uses two media access control (MAC) messages: GATE and REPORT.[1] GATE message is sent from the OLT to an ONU and

Manuscript received September 1, 2003; revised December 1, 2003.

G. Kramer, B. Mukherjee, N. K. Singhal, and A. Banerjee are with the University of California, Davis, CA 95616 USA (e-mail: kramer@cs.ucdavis.edu; mukherjee@cs.ucdavis.edu; singhaln@cs.ucdavis.edu; banerjea@cs.ucdavis.edu).

S. Dixit and Y. Ye are with the Nokia Research Center, Burlington, MA 01803 USA (e-mail: sudhir.dixit@nokia.com; yinghua.ye@nokia.com).

Digital Object Identifier 10.1109/JSAC.2004.830473

[1]Additional messages defined by the MPCP are used by the initialization process.

0733-8716/04$20.00 © 2004 IEEE

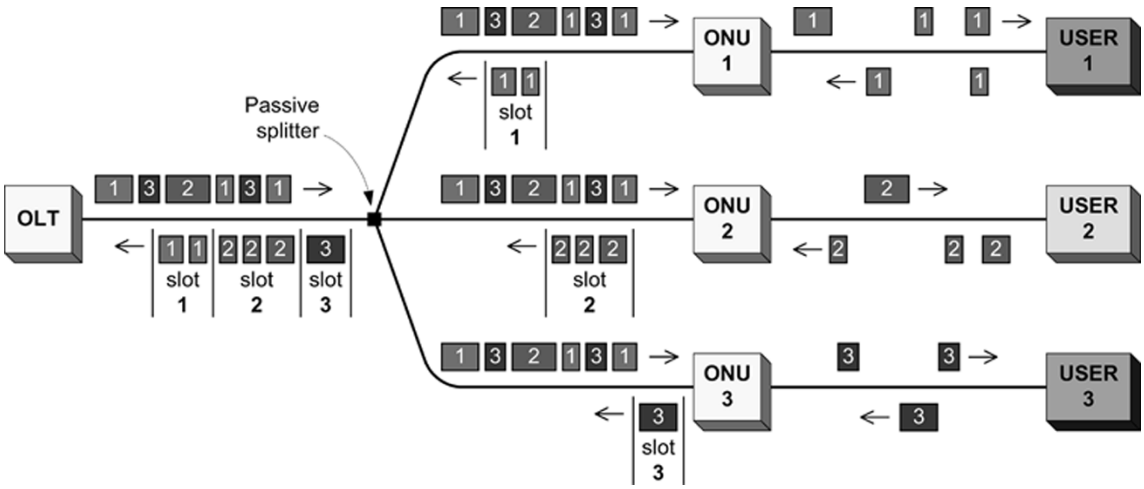



Fig. 1. Upstream and downstream transmissions in EPON.

is used to assign a time slot to the ONU. The REPORT message is sent from an ONU to the OLT to request a next time slot of specific size. Each message is a standard 64-byte MAC Control frame. MPCP is only a message-exchange protocol; it does not specify any algorithm for bandwidth allocation.

In a remote scheduling system like EPON, queues (consumers) could be physically located at a large distance from the centralized scheduler (server) and from one another. This leads to several often-ignored system properties such as *significant queue switch-over overhead*, *large control-plane propagation delay*, and *limited control-plane bandwidth*.

*Significant Queue Switch-Over Overhead:* When switching from one ONU to another, the receiver may need some additional time to readjust the gain since the power levels received from ONUs are different. In addition, the ONUs are required to keep lasers turned off between the transmissions (see [2]). Turning a laser on and off is not an instantaneous process and will also contribute to the switch-over overhead. This leads to a significant overhead when switching from one ONU to another. For example, in EPON, the maximum overhead is 2 $\mu$s.[2]

*Large Control-Plane Propagation Delay:* Control-plane delay is negligible for local schedulers (system-in-a-chip architectures or when queues and the scheduler are connected through a back-plane). But, in a remote scheduling system, the physical distances can be large and delay can exceed by many times the packet transmission time. In addition, in systems like EPON, the control messages are in-band and can be transmitted only in a previously assigned time slot. Thus, the control message delay increases even more, now due to waiting for the next time slot to arrive. This results in the scheduler always operating with somewhat outdated information.

*Limited Control-Plane Bandwidth:* Scheduling multiple clients (queues, ONUs) may require a separate control message to be sent periodically from the scheduler located at the OLT to each client (GATE message) and from each client to the scheduler (REPORT message). Increasing the number of clients may give rise to scalability issues when a significant fraction of the total EPON bandwidth is consumed by the control messages.

[2]The 2 $\mu$s time interval includes laser on/off, automatic gain control (AGC), and clock-and-data recovery (CDR) times.

### B. Objectives of a Remote Scheduling Algorithm

In this paper, we consider an application of a remote scheduler in a subscriber access network. In this network, an ONU may serve one or more subscribers and can have one or more queues assigned to each subscriber. Different queues belonging to one subscriber can be used, for example, to serve different classes of traffic (i.e., voice, video, and data) with different quality-of-service (QoS) guarantees. To satisfy the network requirements, a remote scheduler should meet the following objectives.

*Scalability:* The algorithm should support a large number of queues (several hundreds to several thousands). The algorithm should be efficient and scalable with the number of queues, i.e., overhead should not grow significantly with the number of queues served.

*Guarantees:* Unlike enterprise local area networks (LANs), access networks serve noncooperative users; users pay for service and expect to receive their service regardless of the network state or the activities of the other users. Therefore, the network operator must be able to guarantee a minimum bandwidth $B_i^{\mathrm{MIN}}$ to each queue $i$ assuming, of course, that the queue has enough data to send.

*Fairness:* Idle queues should not use any bandwidth. Excess bandwidth $B^{\mathrm{EX}}$ left by idle queues should be redistributed among backlogged queues in a fair manner, i.e., in proportion to weight $\varphi_i$ assigned to each queue ($B_i^{\mathrm{EX}}/\varphi_i = B_j^{\mathrm{EX}}/\varphi_j$). The fairness of bandwidth distribution should be preserved regardless of whether the queues are located in the same ONU or in different ONUs.

### C. Previous Work

Generalized processor sharing (GPS) [3] is an idealistic fluid model supporting fair resource sharing. Many algorithms were derived from GPS to support fair queueing in systems with atomic protocol data units (i.e., nondivisible cells or packets): weighted fair queueing (WFQ) [4], worst-case fair weighted fair queueing ($\mathrm{WF}^2\mathrm{Q}$) [5], virtual-clock fair queueing (VCFQ) [6], self-clocked fair-queueing (SCFQ) [7], start-time weighted fair queueing (STFQ) [8], and many others. These algorithms were shown to distribute excess bandwidth among the queues

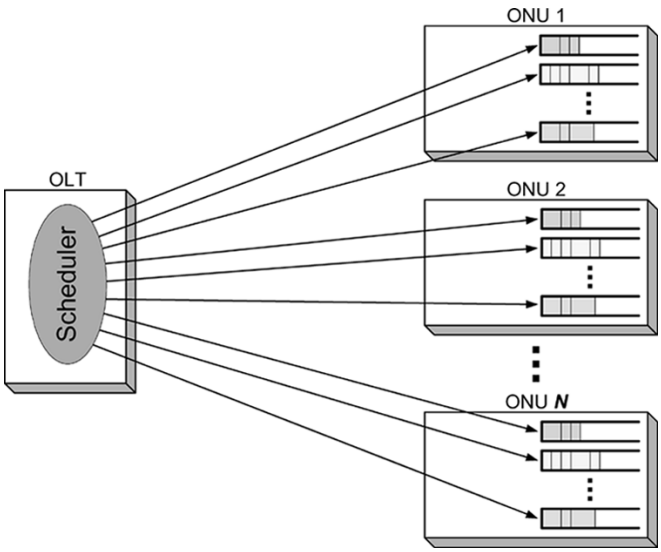



Fig. 2. Flat (single-level) scheduling in EPON.

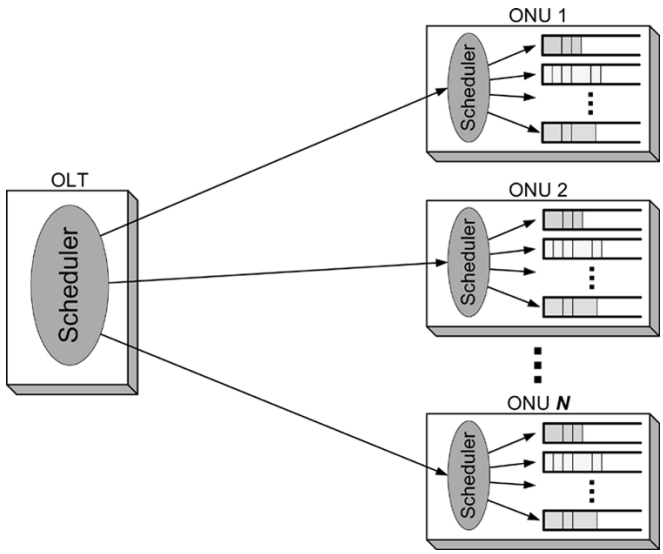



Fig. 3. Hierarchical scheduling in EPON.

almost fairly, i.e., at any moment of time, the amount of service a queue received would differ from an ideal fluid model by at most one maximum-sized data unit (packet or cell). In the following sections, we consider specific characteristics of these and other protocols and their suitability to remote-scheduling systems.

*1) Virtual-Time Versus Cycle-Based Schedulers:* Packet-based GPS (and its derivatives) schedule packets based on their virtual finish time (start time, bin number, etc.). Virtual times depend on packet arrival times and relative weights assigned to queues. This may result in a situation when consecutive packets are sent from different queues located in different ONUs. But this will require a guard time between each pair of such packets. Taking average Ethernet packet size to be $\sim 530$ bytes (measured on a real-access network upstream traffic [9]), EPON line rate of 1 Gb/s, and guard time of 2 $\mu$s, we obtain

$$\text{overhead} = \frac{2\,\mu\text{s}}{\frac{530\text{ bytes}}{1\text{ Gb/s}} + 2\,\mu\text{s}} \approx 32\%.$$

(This calculation assumes that no two consecutively served queues belong to the same ONU.) For smallest Ethernet packet sizes (64 bytes), this overhead can reach 80% (i.e., 80% of bandwidth is wasted on guard time). *Thus, algorithms based on virtual-time in EPON are impractical.*

An alternative solution is to employ a *cycle-based* (also called frame-based) algorithm, where all queues are served consecutively in round-robin fashion. The order of service may be chosen such that all queues belonging to the same ONU are served continuously without a guard band between them. The guard band will only be needed when switching from one ONU to another. A simple extension of round robin in which the service quanta for each queue is proportional to the queue's weight is called weighted round robin (WRR).

*2) Flat (Single-Level) Versus Hierarchical (Multilevel) Schedulers:* Applying a single-level algorithm to an EPON means that a scheduler located in the OLT would individually schedule each consumer (queue) located in multiple ONUs (Fig. 2) such that the required service guarantees are preserved and the excess bandwidth (if any) is fairly divided among backlogged queues.

Scheduling multiple queues requires a separate GATE message to be sent to each queue and a separate REPORT message to be received from each queue. Consider an EPON system with 32 ONUs, 128 subscribers per ONU, and three queues per subscriber, for a total of 12 288 queues. This adds considerable overhead for control messages. For example, assuming that 1/3 of all queues are used for voice traffic with a delay bound of 1.5 ms [10], the OLT should be able to generate 4096 GATE messages within 1.5-ms interval, but at 1-Gb/s EPON rate, it takes 2.75 ms to transmit this many GATE messages (without any data packets); so, 4096 voice queues cannot be supported. Based on this observation, we conclude that, *in a remote scheduling environment, single-level schedulers are nonscalable with the number of queues.*

Several algorithms have been developed to support hierarchical scheduling [hierarchical fair queueing (HFQ) [11], hierarchical round robin (HRR) [12], etc.]. In such schemes, all queues are divided into groups. The high-level scheduler schedules the groups (i.e., provides aggregated bandwidth per group), while the low-level schedulers schedule the queues within each group. The root scheduler treats each group as one consumer and has no information about internal composition of each group. EPON can be naturally divided into a hierarchy of schedulers, where the high-level scheduler is located in the OLT and schedules individual ONUs, and the low-level scheduler is located in each ONU and schedules queues within the ONU (Fig. 3).

In the hierarchical EPON scheduling scheme, the root scheduler (OLT) only schedules the intermediate nodes (ONUs). The OLT would receive one REPORT message from an ONU and would generate one GATE message for the ONU. The GATE and REPORT messages would grant and request an aggregated bandwidth per ONU (i.e., a large time slot which the ONU would internally share among its queues). A hierarchical scheme solves the scalability issue due to elimination of separate GATE and REPORT messages for each queue. It also solves the switch-over overhead issue due to the fact that

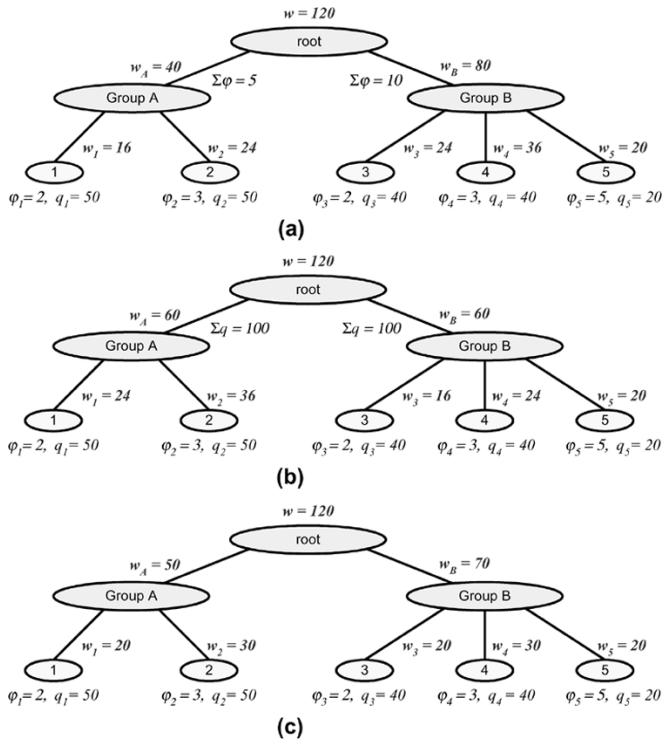



Fig. 4. (a) Sibling-fair scheduling based on cumulative group weight $\Sigma\varphi$. (b) Sibling-fair scheduling based on cumulative group work $\Sigma q$. (c) Cousin-fair scheduling.

all queues in one ONU are served consecutively with no guard times between their transmissions. (Guard times remain only when the OLT switches to serve the next ONU.)

*3) Sibling-Fair Versus Cousin-Fair Schedulers:* The challenging issue with a hierarchical scheme is supporting fair resource distribution among queues in different groups (ONUs). Most hierarchical scheduling protocols known to the authors allow fairness only among *siblings* (i.e., nodes having the same parent). We call such schedulers *sibling-fair* or *locally fair* schedulers. Fig. 4 illustrates bandwidth distribution among 5 queues separated into two groups, $A$ and $B$ ($A = \{1, 2\}$ and $B = \{3, 4, 5\}$). Each queue $i$ is characterized by its weight $\varphi_i$ and its size (unfinished work) $q_i$. The amount of service each queue gets is denoted by $w_i$.

In Fig. 4(a), the scheduler makes its bandwidth-allocation decision based on cumulative weight $\Sigma\varphi$ of each group. Queue 5 has less unfinished work and, thus, requires less service. Unused service left by queue 5 is distributed among its siblings 3 and 4 assuming schedulers at each level are work conserving. It can be easily observed that siblings at each level receive mutually fair service $w$, i.e., for any queues $i$ and $j$ with sufficient amount of unfinished work $q$ ($q_i \geq w_i$), the bandwidth is allocated in proportion to their weights $w_i/\varphi_i = w_j/\varphi_j$. However, the fairness does not extend across multiple groups. For example, we can see that $w_1/\varphi_1 = w_2/\varphi_2$ and $w_3/\varphi_3 = w_4/\varphi_4$, but $w_1/\varphi_1 \neq w_3/\varphi_3$ and $w_2/\varphi_2 \neq w_4/\varphi_4$. Similar outcome is observed if the service is distributed based on any other single value, e.g., the cumulative amount of unfinished work in a group, as shown in Fig. 4(b).

Fig. 4(c) illustrates the desired service distribution that achieves fairness among all leaves with sufficient unfinished work ($w_1/\varphi_1 = w_2/\varphi_2 = w_3/\varphi_3 = w_4/\varphi_4$). We call this scheme a *cousin-fair* (or *globally fair*) scheduling, in contrast to the sibling-fair scheme described above. This scheduler does not provide fairness among intermediate nodes, but allows fairness among all leaves, no matter which group they belong to. Fig. 4(c) shows that the bandwidth allocated to an intermediate node should dynamically change based on the state of all the leaves. While cousin fairness is illustrated here for a two-level system, it is easy to generalize the concept to a hierarchical scheduling system with an arbitrary number of levels.

We note that Rexford *et al.* in [13] reported a dynamic weight-adjustment scheme that allows a hierarchical scheduler to be cousin-fair. In their algorithm, a root-level scheduler receives from each group a cumulative weight as a sum of weights of all nonempty queues in a group, i.e., it is a scheme shown in Fig. 4(a). As soon as a busy queue becomes empty or an empty queue becomes busy, the root scheduler should learn the new weight. Thus, this algorithm is only suitable for systems with a small propagation delay and not for remote scheduling systems.

Summarizing the above survey of scheduling mechanisms, we conclude that, in order to satisfy the objectives stated in Section I-B, the scheduling algorithm should be *hierarchical*, *cycle-based*, and *cousin-fair*. To the best of our knowledge, no such algorithm has been described in the literature yet.

This paper is organized as follows. Section II gives a formal definition of fair scheduling for the case where the queues are allocated guaranteed minimum bandwidth and are assigned weights for sharing the excess bandwidth. In Section III, we present the framework of FQSE suitable for an idealized fluid-network model and discuss the tradeoff between complexity and resource-allocation efficiency (utilization). In Section IV, we introduce additional mechanisms necessary to adopt FQSE for a network based on variable-sized packets. Specifically, we consider a subscriber access network based on EPON. In Section V, we analyze the performance of FQSE, illustrate its cousin fairness, and derive bounds on fairness (often called fairness index). We verify our analytical results using simulation experiments. Section VI concludes this paper.

## II. Fair-Share Remote Scheduling System—A Formal Definition

As stated in Section I-B, the objectives of a remote scheduler is to guarantee minimum service $B^{\mathrm{MIN}}$ to each queue and fairly share the excess service $B^{\mathrm{EX}}$. Typically, bandwidth is distributed in time slots, i.e., a time slot (transmission window) of size $W$ bytes is given to a queue once every $T$ seconds ($T$ is called the *cycle time*). Thus, it is convenient to define minimum (guaranteed) time slot size $W_i^{\mathrm{MIN}}$ that should be given to a queue $i$ to guarantee its minimum bandwidth $B_i^{\mathrm{MIN}}$ ($W_i^{\mathrm{MIN}} = B_i^{\mathrm{MIN}} T$). We also define $W^{\mathrm{CYCLE}}$ as the total available service in one cycle time $T$ (i.e., the number of bytes that can be transmitted in time $T$). Clearly, to guarantee minimum bandwidth, the sum of all $W_i^{\mathrm{MIN}}$ should not exceed $W^{\mathrm{CYCLE}}$. The actual minimum slot size that a queue $i$ gets in cycle $k$ is $w_{i,k}^{\mathrm{MIN}}$

$$w_{i,k}^{\mathrm{MIN}} = \min \begin{cases} q_{i,k} \\ W_i^{\mathrm{MIN}} \end{cases} \tag{1}$$

where $q_{i,k}$ is the length of queue $i$ at the beginning of cycle $k$. Equation (1) states that a queue should never be given a slot larger than the amount of data the queue has accumulated. Total remaining transmission-window size (excess bandwidth) $w_k^{\mathrm{EX}}$ left in cycle $k$ after assigning all minimum slots to all the queues is equal to

$$w_k^{\mathrm{EX}} = W^{\mathrm{CYCLE}} - \sum_{i=1}^{N} w_{i,k}^{\mathrm{MIN}}. \tag{2}$$

We define a backlogged queue to be a queue which cannot or will not be served to exhaustion in one cycle (one queue transmission). The set of all queues backlogged in cycle $k$ is denoted by $\Omega_k$. Each queue $i$ that remains backlogged after serving $w_{i,k}^{\mathrm{MIN}}$ bytes (i.e., with $q_{i,k} > w_{i,k}^{\mathrm{MIN}}$) should get a share of the excess bandwidth $w_k^{\mathrm{EX}}$ proportional to its weight $\varphi_i$, i.e.,

$$w_{i,k}^{\mathrm{EX}} = w_k^{\mathrm{EX}} \frac{\varphi_i}{\sum_{j\in\Omega_k} \varphi_j}. \tag{3}$$

The subtle problem with the definition in (3) arises due to the fact that a queue shall not be given more slot size than it has data to transmit. Thus, if the guaranteed slot $w_{i,k}^{\mathrm{MIN}}$ together with excess slot $w_{i,k}^{\mathrm{EX}}$ exceeds the queue length $q_{i,k}$, the queue will be given a slot size equal to $q_{i,k}$. This means that queue $i$ will be served to exhaustion and, thus, it should not be considered a backlogged queue any more. Removing queue $i$ from the set $\Omega_k$ of backlogged queues will affect the amount of remaining excess bandwidth, as well as the share of each queue that remains backlogged. To capture this effect, we amend (3) as follows:

$$w_{i,k}^{\mathrm{EX}} = \begin{cases} q_{i,k} - w_{i,k}^{\mathrm{MIN}}, & i \notin \Omega_k \\ \left(W^{\mathrm{CYCLE}} - \sum_{j\in\Omega_k} W_i^{\mathrm{MIN}} - \sum_{j\notin\Omega_k} q_{j,k}\right) \frac{\varphi_i}{\sum_{j\in\Omega_k} \varphi_j}, & i \in \Omega_k \end{cases}. \tag{4}$$

Equation (4) says that the excess slot size given to a queue will either be just enough to serve the queue to exhaustion (if queue $i$ does not belong to a set of backlogged queues); otherwise, it will be served in proportion to the queue's weight $\varphi_i$ and the total number of bytes remaining available after serving to exhaustion all nonbacklogged queues and assigning the minimum guaranteed time slots to all backlogged queues. It is important to understand that this is a recursive definition, since the membership in set $\Omega_k$ is determined as: $i \in \Omega_k$ iff $w_{i,k}^{\mathrm{MIN}} + w_{i,k}^{\mathrm{EX}} < q_{i,k}$.

Summing (1) and (4), we get the total time slot size $w_{i,k}$ given to a queue $i$ in cycle $k$ to be shown in (5) at the bottom of the page.

Finally, the cumulative size of all slots assigned in one cycle cannot exceed the cycle capacity $W^{\mathrm{CYCLE}}$. The cumulative slot size also cannot exceed the sum of all queue lengths (i.e., in the case when all the queues can be served to exhaustion in one cycle). This is reflected in (6)

$$\sum_{i=1}^{N} w_{i,k} = \min \begin{cases} \sum_{i=1}^{N} q_{i,k} \\ W^{\mathrm{CYCLE}} \end{cases}. \tag{6}$$

It is easy to verify that (5) summed for all queues indeed complies with the requirement in (6).

A solution to a system of equations described by (5) constitutes a valid schedule compliant with the requirements for guaranteeing the minimum bandwidth and fairly sharing the excess bandwidth.

By specifying the minimum bandwidth $B^{\mathrm{MIN}}$ (or minimum time slot $W^{\mathrm{MIN}}$) and the weight $\varphi$ per connection, the network operator can provision different types of services to subscribers (queues). Table I presents some examples of specifying different services.

## III. Fair Queueing With Service Envelopes (FQSE)

FQSE is a hierarchical remote-scheduling algorithm that distributes service in accordance with (5). The algorithm is based on a concept of a service envelope (SE). A service envelope represents the amount of service (time slot size) given to a node as a function of some nonnegative value which we call satisfiability parameter (SP). SP is a measure of how much the demand for bandwidth can be satisfied for a given node.

In a scheduling hierarchy, each node has its associated SE function. We distinguish the construction of a service envelope for a leaf (denoted $E^*$) from the construction of a service envelope for a nonleaf node (denoted $E$).

$E^*$ is a piecewise-linear function consisting of at most two segments (see Fig. 5, plots $E_2^*$ and $E_3^*$). The first segment begins at a point with coordinate (0, $w_{i,k}^{\mathrm{MIN}}$) and ends at $((q_{i,k} - w_{i,k}^{\mathrm{MIN}})/\varphi_i, q_{i,k})$. The ending SP value is chosen such that the slope of the first segment is exactly $\varphi_i$. The second segment has a slope equal to 0 and continues to infinity.

Intuitively, the meaning of the $E^*$ function should be clear: as the satisfiability parameter changes, the $E^*$ function determines the fair time slot size for the given queue. In the worst case (i.e., when $SP = 0$), an exact $w_{i,k}^{\mathrm{MIN}}$-byte time slot will be given to the queue (i.e., the queue will get its guaranteed minimum service). As satisfiability parameter $s$ increases, the queue will be given an additional time slot (excess bandwidth) equal to $\varphi_i s$. When the time slot size reaches $q_{i,k}$ (total queue length), it will not increase any more, even if $s$ increases. In case a queue has less data than its guaranteed slot size (i.e., $q_{i,k} < W_i^{\mathrm{MIN}}$), the $E^*$ function will consist of only one segment with slope zero (Fig. 5, plot $E_1^*$).

$$w_{i,k} = w_{i,k}^{\mathrm{MIN}} + w_{i,k}^{\mathrm{EX}} = \begin{cases} q_{i,k}, & i \notin \Omega_k \\ W_i^{\mathrm{MIN}} + \left(W^{\mathrm{CYCLE}} - \sum_{j\in\Omega_k} W_i^{\mathrm{MIN}} - \sum_{j\notin\Omega_k} q_{j,k}\right) \frac{\varphi_i}{\sum_{j\in\Omega_k} \varphi_j}, & i \in \Omega_k \end{cases} \tag{5}$$

TABLE I
EXAMPLES OF QUEUE CONFIGURATIONS

| Queue | $B_{MIN}$ | $\varphi$ | Description |
|---|---|---|---|
| 1 | 0 | 2 | This is a best-effort service (no guaranteed bandwidth). Under very heavy network load, this queue may get no service. |
| 2 | 0 | 1 | This is also a best-effort service. If the network load is not heavy, i.e., if excess bandwidth is available, this queue would get half the bandwidth queue 1 gets (consider it a *good-effort* queue). |
| 3 | 10 Mbps | 0 | This queue will never get any extra bandwidth, but it will always get its guaranteed bandwidth. No matter what the network load is, this queue will be able to transmit 10 Mbps of data. This configuration is good for circuit-emulation services. |
| 4 | 10 Mbps | 1 | This queue will always have its guaranteed bandwidth (10 Mbps) plus it will get an excess bandwidth if the network load is not high. When some excess bandwidth is available, this queue would be able to transmit exactly what queue 2 transmits plus 10 Mbps. |

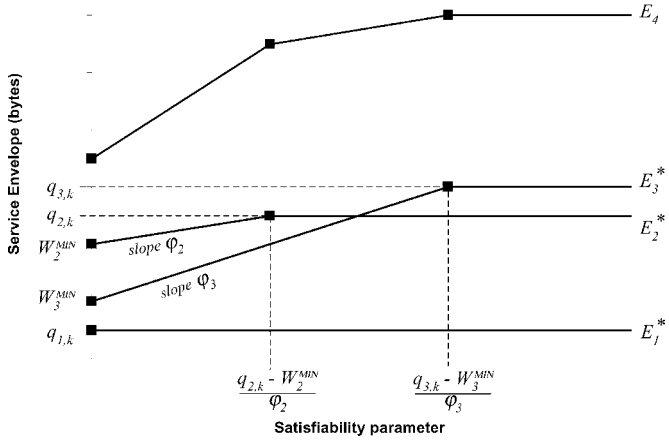



Fig. 5. Construction of service envelopes.

The service envelope $E_i$ of a nonleaf node $i$ is built as a sum of service envelopes of all the node's children

$$E_{i,k} = \sum_{j \in D_i} E_{j,k} \tag{7}$$

where $D_i$ is a set containing all children of node $i$. This is illustrated in Fig. 5, plot $E_4$.

The FQSE algorithm consists of alternating *requesting* and *granting* phases. The following are the steps of the algorithm.

### A. Phase 1–Requesting Service

At the end of transmission in a previously assigned time slot, a node should generate a new service envelope and send it to its parent in a request message. After collecting service envelopes from all its children, an intermediate node would generate its own service envelope by summing all received envelopes and send it to its parent.

A request message typically has a fixed format and may contain at most $K$ point coordinates representing knots of the piecewise-linear service envelope. Since the number of children of any intermediate node $i$ can be arbitrarily large, the service envelope of node $i$ may contain an arbitrarily large number of points. In case that the actual number of points $m_{i,k} > K$, node $i$ will perform a piecewise linear approximation of the function $E_{i,k}$ such that the $m_{i,k}$-point function is described with only $K$ points and can be transmitted in one request message. (The approximation procedure is described in Section III-C.) The approximated function is denoted $\widetilde{E}_{i,k}$. Thus, (7) can be rewritten as follows:

$$E_{i,k} = \sum_{j \in D_i} \widetilde{E}_{j,k} \tag{8}$$

where $\widetilde{E}_{j,k}$ is the approximated service envelope of the $j$th child of node $i$ in cycle $k$.

The requesting phase ends when the root node receives service envelopes from all its children and calculates its own service envelope $E_{0,k} = \sum_{j \in D_0} \widetilde{E}_{j,k}$.

### B. Phase 2–Granting Service

The root node knows the total number of bytes that can be transmitted in one cycle ($W^{\mathrm{CYCLE}}$). When the root scheduler obtains the $E_{0,k}$ function in cycle $k$, it calculates the satisfiability parameter $s_k$ by solving $E_{0,k}(s_k) = W^{\mathrm{CYCLE}}$. Knowing the cycle start time and the satisfiability parameter $s_k$, the root node calculates the time slot start time $t_{j,k}$ for each child $j \in D_0$ such that transmissions from each child do not overlap. This calculation is performed by a procedure $\mathrm{PROCESS\_GRANT}(t_{i,k}, s_k)$ shown in Fig. 6. The time slot start time $t_{j,k}$ and the satisfiability parameter $s_k$ are then transmitted to each child $j$ in a grant message.[3]

Upon receiving the grant message, each intermediate node invokes the same $\mathrm{PROCESS\_GRANT}(t_{i,k}, s_k)$ procedure to further subdivide the time slot among its children.

The granting phase ends with each leaf node receiving the grant message. When leaf node $i$ receives the grant message containing the time slot start time $t_{i,k}$ and the satisfiability parameter $s_k$, it will calculate its own time slot size $w_{i,k} = E^*_{i,k}(s_k)$. When the local clock in the leaf node reaches

[3]The time slot start time in a grant message should be "precompensated" for the round-trip propagation delay as explained in [1].

```
PROCESS_GRANT( t_{i,k}, s_k )

   t = t_{i,k}
   for each node j in D_i
   {
      t_{j,k} = t
      send grant (t_{j,k}, s_k ) to node j

      // calculate transmission time
      // needed to transmit Ẽ_{i,k}(s_k) bytes

      tx_time = time( Ẽ_{i,k}(s_k) )
      t = t + tx_time
   }
```

Fig. 6. PROCESS_GRANT procedure calculates start times for all children of node $i$.

time $t_{i,k}$, the leaf node starts transmission and transmits $w_{i,k}$ bytes of data.[4]

### C. Service Envelope Approximation Schemes

In the requesting phase (phase 1), each intermediate node $i$ would collect service envelopes from all its children and create service envelope $E_{i,k} = \sum_{j \in D_i} \widetilde{E}_{j,k}$, which it will send to its parent in a Request control message. A Request message can only accommodate a fixed number of points $K$. If node $i$ has $|D_i|$ children, the $E_{i,k}$ function may have $m_{i,k}$ points: $1 \leq m_{i,k} \leq |D_i| \times (K-1) + 1$ (the first point always has $\mathrm{SP} = 0$ and will coincide for all children). If the actual number of points $m_{i,k} > K$, node $i$ will perform a piecewise-linear approximation of the $E_{i,k}$ function such that the $m_{i,k}$-point SE function is described with only $K$ points. We denote the approximated function by $\widetilde{E}_{i,k}$.

In performing this approximation, we set our objective at minimizing the maximum error (minimize $\max_{\forall s} |\widetilde{E}_{i,k}(s) - E_{i,k}(s)|$). We also require that the error is nonnegative ($\widetilde{E}_{i,k}(s) \geq E_{i,k}(s), \forall s$). It is easy to see that allowing a negative error would mean that a time slot granted by a parent to a child ($\widetilde{E}_{i,k}(s)$) could be smaller than the time slot assumed by the child ($E_{i,k}(s)$). This may cause a collision with the data transmitted by some other child of parent node. Keeping the approximation error nonnegative will at most increase the dead zone between two adjacent time slots (i.e., it may reduce the channel's utilization), but it will ensure that each node can get its fair slot size and no data collisions will occur due to slot overlaps.

The approximation procedure employs two functions: $\mathrm{CONSTRUCT\_APPROX}(E, \varepsilon)$ and $\mathrm{FIND\_APPROX}()$. Given a fixed maximum error $\varepsilon$ and the original service envelope $E$, the $\mathrm{CONSTRUCT\_APPROX}(E, \varepsilon)$ function constructs an approximated service envelope $\widetilde{E}$, which has the minimum number of points. This procedure returns SUCCESS if the approximated envelope can be constructed with $K$ or less points, and FAILURE, otherwise.

[4]The actual transmission size may be less than $w_{i,k}$ bytes for the case of packet-based networks due to nondivisible packets not filling the time slot completely. The necessary adaptation mechanisms for variable-size-packet-based schedulers are discussed in Section IV.

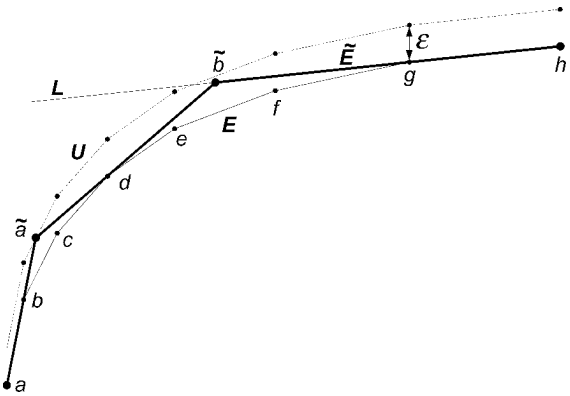



Fig. 7. Approximating the service envelope $E$.

Function $\mathrm{FIND\_APPROX}()$ performs binary search on $\varepsilon$ invoking CONSTRUCT_APPROX at each step. This function will stop when it finds $\varepsilon'$ such that $\mathrm{CONSTRUCT\_APPROX}(E, \varepsilon') = \mathrm{TRUE}$, but $\mathrm{CONSTRUCT\_APPROX}(E, \varepsilon - \Delta\varepsilon) = \mathrm{FALSE}$, where $\Delta\varepsilon$ is the smallest error increment (some small constant).

$\mathrm{CONSTRUCT\_APPROX}(E, \varepsilon)$ constructs an approximated piecewise-linear enevelope $\widetilde{E}$, as explained in the following steps.

- Step 1) Construct a piecewise linear curve $U$ (Fig. 7), all points of which have an error $\varepsilon$ from the original service envelope.
- Step 2) Draw a line $L$ which coincides with the last segment of the original envelope $E$ (segment $(gh)$ in Fig. 7).
- Step 3) Extend the first segment $(ab)$ until it intersects the curve $U$ or the line $L$, whichever appears first. Call the point of intersection $\widetilde{a}$.
- Step 4) From $\widetilde{a}$, draw a tangent to the original service envelope $E$ so that it intersects $E$ at point $d$. (It may happen that the tangent has the same slope as one of the segments, in which case the tangent intersects $E$ at multiple points.)
- Step 5) Extend the tangent $(\widetilde{a}d)$ until it intersects the curve $U$ or the line $L$ again. Call the new intersection point $\widetilde{b}$.
- Step 6) Repeat steps 4 and 5 for each new point of intersection with curve $U$ or line $L$ until $K$ such new points are found or the last point of the original curve $E$ (point $h$ in the example on Fig. 7) is reached.

In [14], we proved that slopes $\tau$ of segments of any service envelope are decreasing, i.e., for any segments $i$ and $j$, $\tau_i > \tau_j$ iff $i < j$. Relying on this property of service envelopes, the test of whether line $(\widetilde{a}d)$ is tangential to $E$ can be performed in $O(1)$ time. Indeed, since the segment slopes are strictly decreasing; line $(\widetilde{a}d)$ will be tangential to $E$ if and only if the slope of $(cd)$ is larger than or equal to the slope of $(\widetilde{a}d)$ and the slope of $(de)$ is smaller than or equal to the slope of $(\widetilde{a}d)$.

Each point of the original service envelope $E$ is passed only once, whether while searching for a tangential line, or searching for the intersection of the tangential line with the curve $U$.

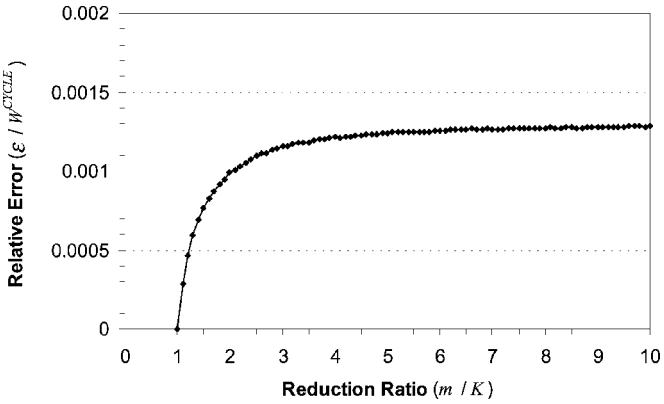



Fig. 8. Relative approximation error.

Therefore, CONSTRUCT_APPROX runs in $O(m)$, where $m$ is the number of points in curve $E$ as defined earlier.[5]

Finally, FIND_APPROX() procedure invokes CONSTRUCT_APPROX at most $O(\log \varepsilon^{\mathrm{MAX}})$ times, where $\varepsilon^{\mathrm{MAX}}$ is the maximum error possible for any service envelope. Because only $W^{\mathrm{CYCLE}}$ bytes can be granted in one cycle, no service envelope needs to have any points with envelope value exceeding $W^{\mathrm{CYCLE}}$.[6] Thus, no approximation error can exceed $W^{\mathrm{CYCLE}}$ (i.e., $\varepsilon^{\mathrm{MAX}} \leq W^{\mathrm{CYCLE}}$). To summarize, the running time of the entire approximation procedure is $O(m \log W^{\mathrm{CYCLE}})$. Pseudocode implementations of CONSTRUCT_APPROX (Fig. 16) and FIND_APPROX (Fig. 18) functions are shown in Appendix A.

We measured the approximation error using a simulation experiments with a large number of randomly generated service envelopes. Fig. 8 presents the average relative approximation error (measured as $\varepsilon/W^{\mathrm{CYCLE}}$) for different reduction ratios ($m_{i,k}/K$). Each point on a plot represents the average error measured over 10 000 service envelopes.

It can be seen that the average approximation error stabilizes at 0.001 28, i.e., we can expect each intermediate node to introduce 0.128% overhead. While this overhead is reasonably small for one node, it may accumulate for multiple nodes. Thus, the scheduling hierarchy should be designed in such a way as to keep the number of intermediate nodes small.

### D. FQSE Complexity

In the requesting phase, each node $i$ should perform two operations: 1) obtain service envelope by summing service envelopes received from all its children and 2) perform approximation, if necessary.

Each envelope received from a child may contain at most $K$ points and is sorted by satisfiability parameter $s$. Thus, to calculate its service envelope, node $i$ should first merge points of received envelopes together, and then it should calculate the cumulative envelope values at each point. Performing pair-wise merging, node $i$ would first merge $|D_i|/2$ pairs of $K$-point envelopes resulting in $|D_i|/2$ $2K$-point envelopes. In the next iteration, these $|D_i|/2$ envelopes will be merged together resulting in $|D_i|/2$ $4K$-point envelopes. Node $i$ will continue merging until, after $\log_2 |D_i|$ steps, the last pair is merged into one $|D_i|K$-point envelope. Therefore, the complexity of this operation is

$$O\left(2K\frac{|D_i|}{2}+4K\frac{|D_i|}{4}+\cdots+K|D_i|\right)=O(K|D_i|\log(|D_i|)).$$

In the following step, node $i$ may need to perform an approximation procedure. As was shown in Section III-C, the complexity of this operation is bounded by $O(K|D_i|\log W^{\mathrm{CYCLE}})$(since $m_i = K|D_i|$). Thus, the overall complexity of the requesting phase at each node is bounded by $O(K|D_i|(\log|D_i| + \log W^{\mathrm{CYCLE}}))$.

In the granting phase, each node invokes PROCESS_GRANT() procedure (Fig. 6) to send a Grant message to each child; therefore, the total work in the granting phase is $O(|D_i|)$.

### E. Granting Schemes

The FQSE algorithm requires the root scheduler to receive service envelopes from all its children before calculating the satisfiability parameter for the next cycle. Each intermediate node should also receive the envelopes from all the children before generating its own envelope. Fig. 9 illustrates this granting scheme for the scheduling hierarchy shown in Fig. 4. The obvious drawback of this scheme is that each cycle will incur an overhead equal to the maximum round-trip delay plus message processing delay at each level in the hierarchy.

In an alternative approach, the root node may separate its children into two or more groups and schedule each group independently. Fig. 10 presents an example of dividing all the nodes into two groups: A and B. The root schedules nodes from group A, while collecting requests from group B. Then, when all the requests from group B are collected, the root would schedule all nodes from group B, while receiving requests from group A. This scheme is free from the overhead shown in Fig. 9, but it provides fairness only among queues within each of the two groups.

The impact of the fact that fairness is only provided within each group can be lessened by carefully grouping the queues. For example, some service-level agreements (SLA) may only require fixed guaranteed bandwidth and no excess bandwidth. Such queues do not participate in excess bandwidth sharing and, therefore, are good candidates for being grouped together. The rest of the queues could be placed in the other group, and they will get a fair share of the excess bandwidth. Another situation when grouping may become even more beneficial is when different types of customers are served by the same subscriber access network, e.g., business and residential subscribers may be separated into different groups.

## IV. FQSE Adaptation for EPON

So far, in our description, we assumed a fluid network model in which the transmission quanta can be infinitesimally small. The time slot assignment guarantees fairness in term of raw bandwidth, i.e., when an entire time slot can be utilized if there is data available in the corresponding queue. This FQSE algorithm

[5]In [14], we have additionally considered an implementation of CONSTRUCT_APPROX performing binary search over the points of the original service envelope to locate the tangent line and its intersections. This version runs in $O(K \log m)$ time.

[6]If, after summing all the envelopes received from the children, some points have envelope values above $W^{\mathrm{CYCLE}}$, such points should be pruned from the resulting envelope.

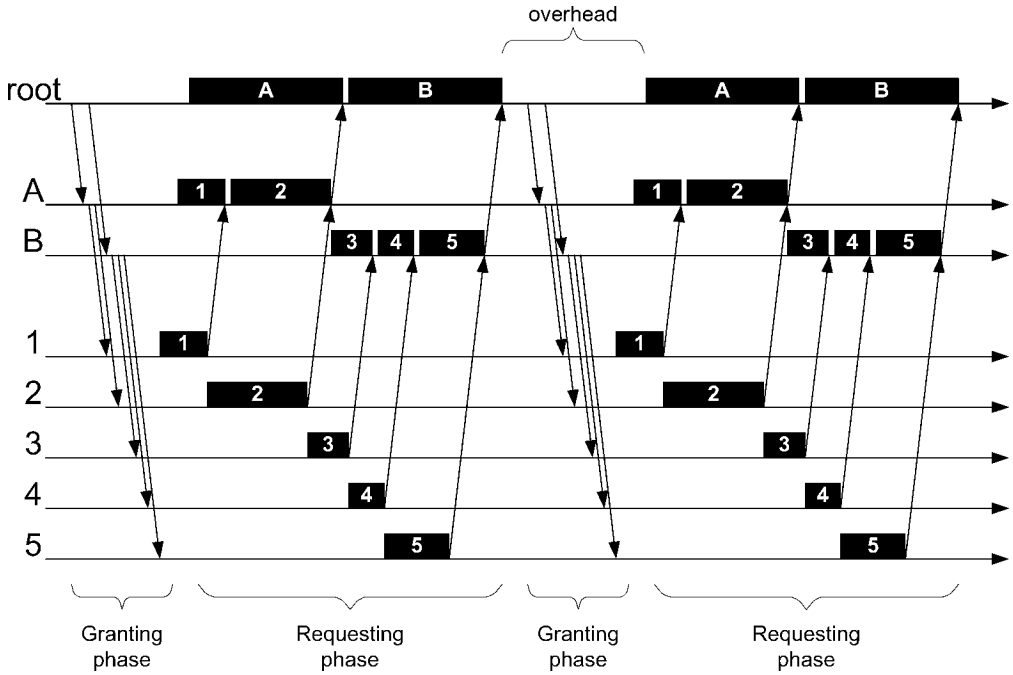



Fig. 9. Collecting all requests before scheduling grants.

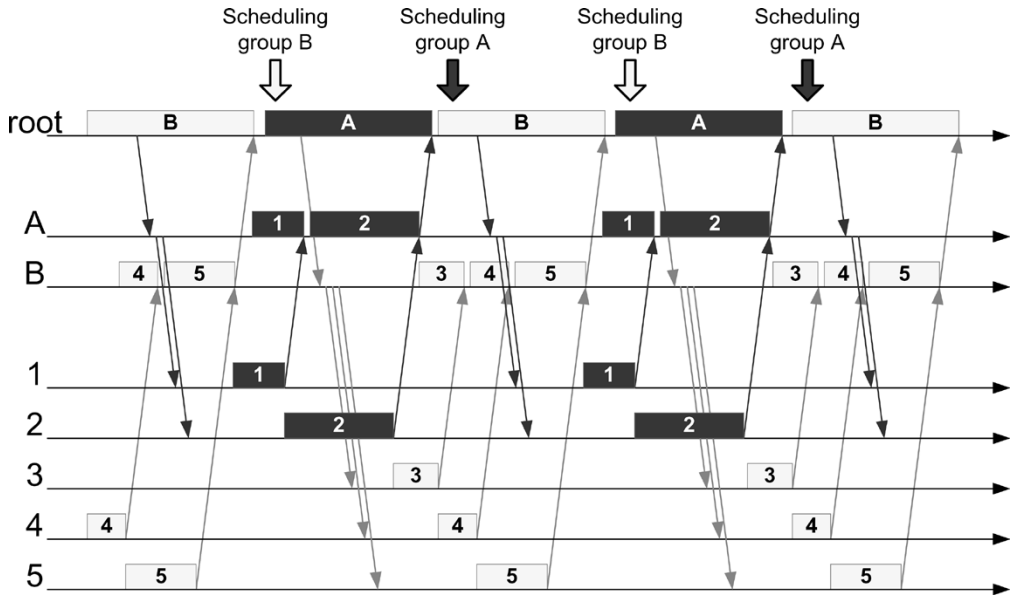



Fig. 10. Scheduling two groups of nodes separately.

can be easily adapted for asynchronous transfer mode (ATM) traffic by simply measuring the time slot size in units of ATM cells (53 bytes). In EPON, however, we are dealing with indivisible packets of variable sizes. Ethernet packets cannot be fragmented; therefore, if the head-of-line (HOL) packet does not fit in the remaining time slot, it will be deferred to the next time slot, while the current time slot will have an unused remainder [15]. This creates two additional issues that the algorithm must address: *HOL blocking* and *bandwidth utilization*.

### A. HOL Blocking

HOL blocking is a result of coupling between bandwidth and latency in a time-sharing packet-based system. It is best explained by an example. Consider the case when a connection should be provisioned a guaranteed bandwidth of 1 Mb/s (and no excess bandwidth) and latency $\leq$ 1 ms (i.e., cycle time is 1 ms). This connection should be given a fixed time slot of size 1 Mb/s $\times$ 1 ms $=$ 125 bytes. All Ethernet packets exceeding the 125-byte size will be blocked in this case. Increasing the time slot size would require a larger cycle time in order to keep the connection bandwidth at a fixed value of 1 Mb/s. However, larger cycle times will violate latency requirements.

To resolve the HOL blocking problem, we allow a connection to occasionally request a minimum slot size larger than its guaranteed minimum slot size $W^{\mathrm{MIN}}$. This approach may lead to a loss of *short-term fairness*, when, in one cycle, a queue may be given a larger slot to accommodate a larger packet. To account

for overused bandwidth (i.e., to maintain long-term fairness), we introduce a per-queue counter called *overdraft*. Overdraft of queue $i$ at the beginning of cycle $k$ (denoted $v_{i,k}$) is estimated as

$$v_{i,k} = v_{i,k-1} + W_{i,k-1}^{\mathrm{MIN}} - W_i^{\mathrm{MIN}} \tag{9}$$

where $W_{i,k}^{\mathrm{MIN}} =$ minimum time slot size requested by queue $i$ in cycle $k$ and $W_i^{\mathrm{MIN}} =$ nominal minimum time slot ($W_i^{\mathrm{MIN}} = B_i^{\mathrm{MIN}} T$). A positive overdraft value means that the queue consumed more service than it is entitled to. Denoting $S_{i,k}^{\mathrm{HOL}} =$ size of HOL packet in queue $i$ at the beginning of $k$th cycle, we calculate $W_{i,k}^{\mathrm{MIN}}$ as follows:

$$W_{i,k}^{\mathrm{MIN}} = \begin{cases} \max\{S_{i,k}^{\mathrm{HOL}}, W_i^{\mathrm{MIN}}\}, & v_{i,k} \le 0 \\ 0, & v_{i,k} > 0 \end{cases}. \tag{10}$$

Equation (10) says that, if no excess service was received by an ONU before the $k$th cycle, the ONU may request a larger time slot in the $k$th cycle to accommodate a large HOL packet. This, of course, will be counted as a service overdraft ($v_{i,k} > 0$), and, in the following few cycles, the queue may be reverse-compensated by receiving less service, until overdraft becomes less than or equal to zero. At this point, if the next HOL packet exceeds $W_i^{\mathrm{MIN}}$, the queue will get excess service again.[7]

We analyze the effects of loss of short-term fairness due to the HOL-blocking avoidance mechanism in Section V-D.

### B. Bandwidth (Time Slot) Utilization

Ethernet traffic consists of nondivisible packets of variable sizes. In most cases, these packets cannot fill the slot completely (i.e., packet delineation in a buffer does not match slot size). This leads to an unused slot remainder and decreased bandwidth utilization. In [15], we derived a formula for the estimated size of the remainder for an arbitrary packet-size distribution. With the empirical trimodal packet-size distribution reported in [9], the average size of the remainder is 595 bytes.

While each individual queue may be blocked on HOL packet, all the remainders together (assuming there are many queues in an ONU) constitute a considerable chunk of slot space, sufficient for sending several more complete packets. To utilize this bandwidth, we borrow the idea of per-queue *deficit counters* from the deficit round-robin (DRR) algorithm [16]. An unused remainder is added to the queue's deficit. When all queues have transmitted all their frames that fit in their granted slots $w_{i,k}$, the ONU performs a second pass and attempts to transmit the HOL packet from a queue with the highest value of its deficit counter. When a packet is transmitted, the value of the deficit counter is decremented by the size of the transmitted packet. The value of the deficit counter is retained between the cycles and can accumulate if a queue does not get a chance to send additional data to compensate for previously unused remainders. This approach efficiently utilizes the bandwidth, with one remainder left per ONU, rather than one remainder per each queue. Additionally, because an ONU can choose among the HOL packets in all the backlogged queues, the remainder left per ONU is considerably smaller than the one associated with a single first-in–first-out (FIFO) queue. Fig. 11 shows the average size of the unused ONU remainder as a function of the number of backlogged queues in the ONU. It is interesting to notice the effects of multiplexing gain on slot utilization. When there is only one backlogged queue, the average remainder is 595 bytes. When the number of backlogged queues per ONU reaches 16, the average remainder drops to only 40 bytes. Further increase in the number of backlogged queues does not provide any significant improvement.

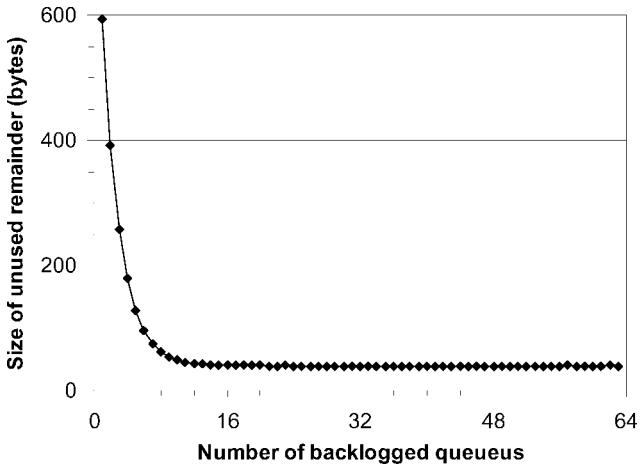



Fig. 11. Size of unused remainder in an ONU.

While using the deficit counter scheme, we should be aware of two problems: *starvation of an old queue* and *starvation of a new queue*. By *new queue*, we mean a queue that became busy after a long idle interval; an *old queue* is a queue that has been busy for a long period of time.

As was explained in [16], idle queues should not accumulate the deficit. Allowing the deficit to accumulate during idle periods would permit a new queue to get an unfairly large bandwidth at the expense of old queues, which will lead to *starvation of old queues*. Thus, the deficit counter remains zero for all idle queues.

One important distinction between FQSE and DRR scheme is that the deficit cannot be completely satisfied in FQSE; no matter how many queues the ONU has, the average unused slot remainder is not zero. In other words, after each transmission cycle, the cumulative unsatisfied deficit would increase by the size of the unused slot remainder (one remainder per ONU). For old queues, this deficit may accumulate for as long as the queues remain busy. When a previously idle queue becomes busy (i.e., a new queue appears), its deficit is much lower than the deficit of a queue that was busy for a long time (due to accumulation of unsatisfied deficit). This could lead to *starvation of a new queue*. To overcome this problem, after each transmission opportunity, all deficit counters will be decreased by the value of the smallest deficit counter among all the busy queues. In other words, we enforce a condition that the "most satisfied" old queue always has $deficit = 0$. Since deficit does not accumulate for idle queues, any new queue will have deficit 0 and, thus, would have the same chance for service as the "most satisfied" old queue.

[7]With a small probability, the cumulative request from all the queues may exceed the cycle capacity $W^{\mathrm{CYCLE}}$. Should this happen, the cycle time must increase to accommodate larger slots. Increased cycle time can affect the accuracy of bandwidth assignment and the SLA guaranteed to a subscriber. In [14], we investigated a way to combat this problem by controlling the guaranteed bandwidth provisioning.

TABLE II
SYSTEM PARAMETERS

| Parameter | Description | Value | Notes |
|---|---|---|---|
| $N$ | Number of ONUs | 16 | Depends on power budget in EPON. Objectives of IEEE 802.3ah task force require at least 16 ONUs (see [17]). |
| $R_U$ | Line rate of user-to ONU link | 100 Mbps | Most LANs and home networks use Fast Ethernet operating at 100 Mbps. |
| $R_N$ | EPON line rate | 1000 Mbps | Objectives of IEEE 802.3ah task force define EPON line rate = 1 Gbps (see [17]). |
| $Q$ | Number of queues per ONU | 64 | |
| $B$ | Buffer size for each queue | 64 Kbyte | |
| $G$ | Guard interval between timeslots | 1 μs | This value should be large enough to allow transmitters to turn laser on and off, and receivers to perform gain adjustment and lock on received clock. |
| $T$ | Cycle time | 1 ms | ITU-T G.114 recommends that, from the overall end-to-end delay budget of 150 ms for a voice data, no more then 1.5 ms be spent in the access network (one-way) [10]. Cycle time of 1 ms will allow *average* data delay of 1.5 ms (in absence of congestion). |

## V. FQSE PERFORMANCE

### A. Experimental Setup

In this paper, we consider an EPON access network consisting of an OLT and $N$ ONUs, each containing $Q$ queues. Propagation delay between each ONU and the OLT is uniformly distributed over the interval [50 $\mu$s, 100 $\mu$s], which corresponds to distances between the OLT and ONUs ranging from 10 to 20 km.

The transmission speed of the EPON and the user access link may not necessarily be the same. In our model, we consider $R_U$ Mb/s to be the data rate of the access link from a user to an ONU, and $R_N$ Mb/s to be the rate of the upstream link from an ONU to the OLT. (Typically, $R_N > R_U$.) Line rates for each link are the same in upstream and downstream directions.

Table II summarizes the parameters used in our simulation experiments.

We designate queues 1 to 4 in ONU A and queues 1 to 4 in ONU B as our test queues (see Fig. 12). The test queues are assigned the guaranteed bandwidth and weight as shown in Table I. The rest of the queues were used to generate background traffic (ambient load). Among these background queues, 18 queues were assigned a guaranteed bandwidth of 1 Mb/s and weight $= 1$, and the remaining queues were best-effort queues (guaranteed bandwidth $= 0$) and weight $= 1$.

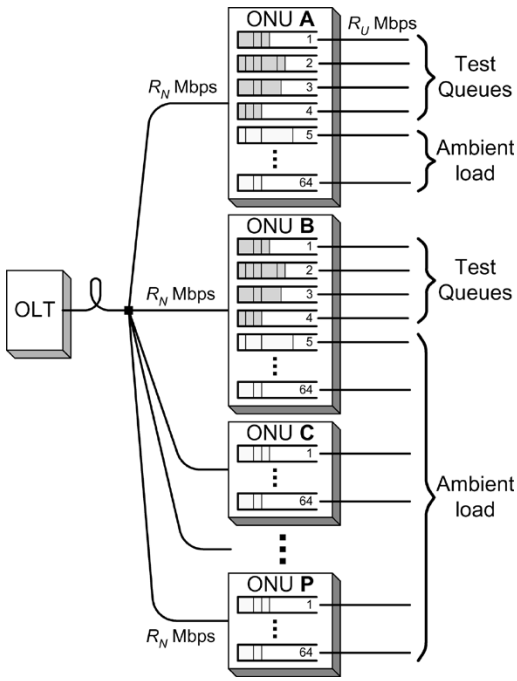



Fig. 12. Experimental EPON model.

### B. Fairness of FQSE

In this section, we analyze the fairness of FQSE by measuring the throughput of the four test queues (1–4) located in one ONU (ONU A). The throughput of each queue was measured over 1-s intervals at four different levels of ambient load (generated by all other queues in all the ONUs).

Each test queue was input a bursty traffic at an average load of 90 Mb/s. Since the FQSE scheduler is work-conserving (i.e., it never grants to any queue a slot larger than the queue length), we expect that the burstiness of the input traffic would be reflected in the queues' throughput. To illustrate the effects of traffic burstiness, we analyzed the queue throughput with two traffic types: short-range dependent (SRD) and long-range dependent (LRD). Both traffic types are bursty (consisting of alternating ON/OFF periods) with burst sizes in SRD traffic having negative exponential distribution, and burst sizes in LRD having a heavy-tail distribution (e.g., Pareto distribution). The LRD traffic was generated using the method described in [18] and was verified to be self-similar with Hurst parameter 0.8.

First, we note that queue 3, which was configured to have 10-Mb/s fixed bandwidth (weight $w_3 = 0$), indeed has a constant throughput regardless of the ambient load (see Fig. 13). The remaining queues were allowed to use the excess bandwidth, if available. In the first 25-s interval, the ambient load was kept relatively low ($\sim$ 180 Mb/s), so that each queue with nonzero weight (queues 1, 2, and 4) was able to send all arrived packets and never became backlogged. The average throughput of each queue was the same as the average load (90 Mb/s). We can see that, in case of LRD traffic [Fig. 13(b)], the throughput

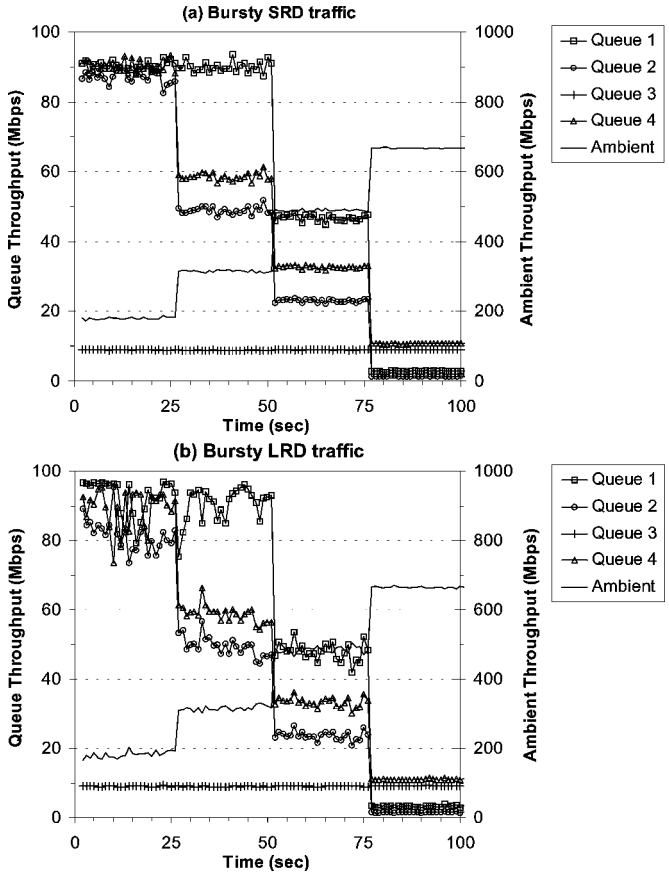



Fig. 13. Throughput of test queues for under different ambient loads.

is bursty (even after averaging over 1-s intervals) reflecting the burstiness of the incoming data stream. In case of SRD traffic [Fig. 13(a)], the averaging effects were much more pronounced and the resulting plots are smoother.

When, at time $t = 25$, the ambient load increases to $\sim 320$ Mb/s, queues 2 and 4 are not able to transmit all the incoming packets and become backlogged. From this moment on, they will maintain the fair relative throughput, with queue 4 always being able to send 10 Mb/s more than queue 2. Queue 1 supposes to have twice the throughput of queue 2. This, however, would give queue 1 more than 90-Mb/s bandwidth, so it only uses 90 Mb/s and does not become backlogged until time $t = 50$. At $t = 50$, when the ambient load increases to $\sim 500$ Mb/s, all four queues become backlogged and all are assigned fair bandwidth. Finally, at time $t = 75$, the ambient load increases even more (to $\sim 660$ Mb/s) and the available excess bandwidth decreases to a very small amount. At this time, the throughput of each queue approaches its guaranteed bandwidth: 10 Mb/s for queues 3 and 4, and zero for queues 1 and 2.

### C. Analysis of Cousin-Fairness

*Claim 1:* FQSE is a cousin-fair scheduling algorithm.

Consider any two nonsibling leaves (queues) $i$ and $j$, and let $E^*_{i,k}$ and $E^*_{j,k}$ be their respective service envelopes in cycle $k$, and let $w_{i,k}$ and $w_{j,k}$ be their time slot sizes given satisfiability parameter $= s$ ($w_{i,k} = E^*_{i,k}(s)$ and $w_{j,k} = E^*_{j,k}(s)$). We want to show that $w_{i,k}$ and $w_{j,k}$ are mutually fair time slot sizes (i.e., excess bandwidth given to each queue is proportional to the queue's weights: $w^{\mathrm{EX}}_{i,k}/\varphi_i = w^{\mathrm{EX}}_{j,k}/\varphi_j$).

*Proof:* The claim is trivially true when the satisfiability parameter $s$ completely satisfies one or both queues (i.e., when $w_{i,k} = q_{i,k}$ and/or $w_{j,k} = q_{j,k}$ ). In this case, one or both queues will be served to exhaustion; the remaining (at most one) backlogged queue can take all the remaining bandwidth and that will be fair.

Let us consider a case when both queues cannot be completely satisfied (i.e., in both functions $E^*_{i,k}$ and $E^*_{j,k}$, the coordinate $s$ belongs to segments with slope $\neq 0$). Excess bandwidths given to queues $i$ and $j$ in this case are $w^{\mathrm{EX}}_{i,k} = w_{i,k} - w^{\mathrm{MIN}}_{i,k}$ and $w^{\mathrm{EX}}_{j,k} = w_{j,k} - w^{\mathrm{MIN}}_{j,k}$. We need to show that $w^{\mathrm{EX}}_{i,k}/\varphi_i = w^{\mathrm{EX}}_{j,k}/\varphi_j$. By construction of the $E^*$ function, we have

$$\frac{w^{\mathrm{EX}}_{i,k}}{\varphi_i} = \frac{w^{\mathrm{EX}}_{j,k}}{\varphi_j}$$
$$\Rightarrow \frac{w_{i,k} - w^{\mathrm{MIN}}_{i,k}}{\varphi_i} = \frac{w_{j,k} - w^{\mathrm{MIN}}_{j,k}}{\varphi_j}$$
$$\Rightarrow \frac{w^{\mathrm{MIN}}_{i,k} + s\varphi_i - w^{\mathrm{MIN}}_{i,k}}{\varphi_i} = \frac{w^{\mathrm{MIN}}_{j,k} + s\varphi_j - w^{\mathrm{MIN}}_{j,k}}{\varphi_j}.$$

Claim 1 holds no matter whether queues $i$ and $j$ have the same parent or not (i.e., this is a cousin-fair time slot allocation).

To verify the property of cousin-fairness experimentally, we compare the throughputs of four test queues located in ONU A with their counterparts (queues having the same guaranteed bandwidth and weight values) in ONU B. For each test queue $i$, we plot the ratio of the throughput of queue $i$ in ONU A to the throughput of queue $i$ in ONU B (Fig. 14).

It can be seen that, for backlogged queues, this ratio approaches 1. For nonbacklogged queues, the ratio may deviate from 1, reflecting the burstiness of the input data stream. This behavior is expected for a work-conserving system.

### D. Analysis of Fairness Bound

In Section IV-A, we explained that the HOL-blocking avoidance mechanism could result in the short-term loss of fairness. In this section, we derive the bound for fairness error and compare it with experimental results. We start with the following claims.

*Claim 2:* For any queue $i$ with $W^{\mathrm{MIN}}_i \leq S^{\mathrm{MAX}}$, the running values of the overdraft counter $v_{i,k}$ are bounded as $1 - W^{\mathrm{MIN}}_i \leq v_{i,k} \leq S^{\mathrm{MAX}} - W^{\mathrm{MIN}}_i$, if $W^{\mathrm{MIN}}_i < S^{\mathrm{MAX}}$, and $v_{i,k} = 0$, if $W^{\mathrm{MIN}}_i \geq S^{\mathrm{MAX}}$.

*Proof:* Equations (9) and (10) can be combined, as shown in (11a)–(11c) at the bottom of the next page.

In (11a), the overdraft value in the next cycle remains the same as it was in the previous cycle. Equation (11b) results in an increased value of $v_{i,k}$. The value of $v_{i,k}$ will be the largest when $v_{i,k-1} = 0$ [by condition (11b), $v_{i,k}$ should be nonpositive] and $S^{\mathrm{HOL}}_{i,k-1} = S^{\mathrm{MAX}}$.

Equation (11c) results in a decreased value of $v_{i,k}$. The value of $v_{i,k}$ will be the lowest when $v_{i,k-1} = 1$ [by condition (11c), $v_{i,k}$ should be positive].

Thus, we have $1 - W^{\mathrm{MIN}}_i \leq v_{i,k} \leq S^{\mathrm{MAX}} - W^{\mathrm{MIN}}_i$.

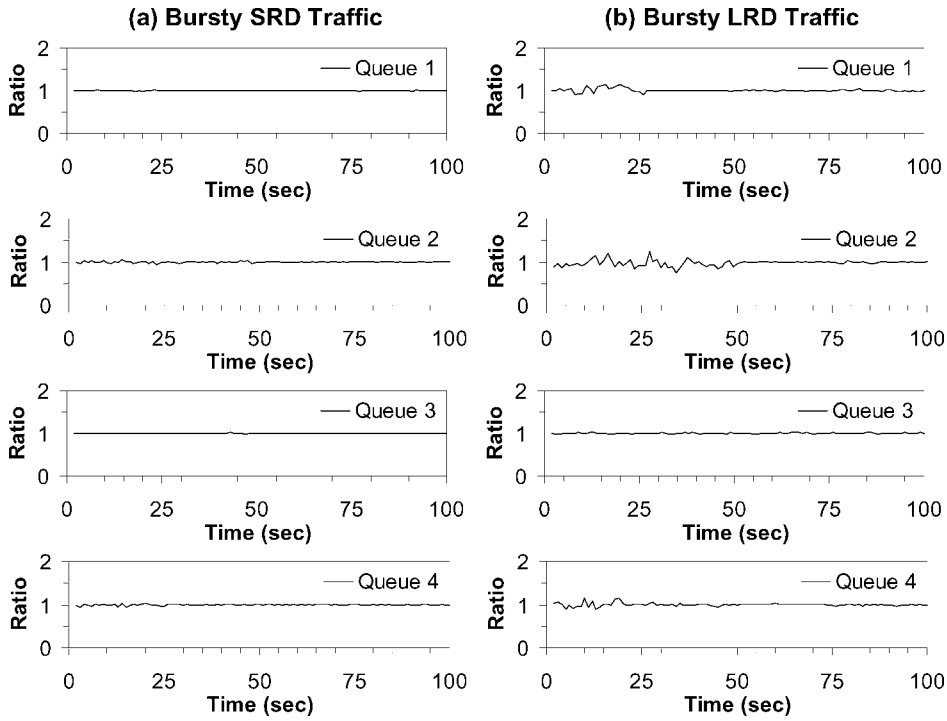



Fig. 14. Ratio of throughputs for queues located in different ONUs.

Also, from (11), we can derive bounds on the increment/decrement that the overdraft counter can have in one cycle.

*Claim 3:* Increment that the overdraft counter may get in one cycle is bounded by $S^{\mathrm{MAX}} - W_i^{\mathrm{MIN}}$, and the decrement is bounded by $W_i^{\mathrm{MIN}}$, i.e., for any $k$

$$-W_i^{\mathrm{MIN}} \le v_{i,k} - v_{i,k-1} \le S^{\mathrm{MAX}} - W_i^{\mathrm{MIN}}.$$

The proof follows directly from (11).

*Definition:* Let $w_i(k,n)$ be the cumulative optimal service [according to (5)] that queue $i$ should receive during $n$ cycles of the scheduler starting in cycle $k$, i.e., $w_i(k,n) = w_{i,k} + w_{i,k+1} + \cdots + w_{i,k+n-1}$; and let $\widetilde{w}_i(k,n)$ be the actual cumulative service received by queue $i$ during $n$ cycles of the scheduler starting in cycle $k$, i.e., $\widetilde{w}_i(k,n) = \widetilde{w}_{i,k} + \widetilde{w}_{i,k+1} + \cdots + \widetilde{w}_{i,k+n-1}$. ($\widetilde{w}_{i,k}$ may not be equal to $w_{i,k}$ because of the HOL-blocking avoidance mechanism, i.e., $W_{i,k}^{\mathrm{MIN}} \neq W_i^{\mathrm{MIN}}$).

*Theorem 1:* Service received by queue $i$ in any interval of $n$ cycles ($n = 1, 2, 3, \ldots$), during which the queue remains backlogged, does not exceed the optimal service by more than $C_i^+(n)$, and does not fall short by more than $C_i^-(n)$, i.e.,

$$C_i^-(n) \le \widetilde{w}_i(k,n) - w_i(k,n) \le C_i^+(n)$$

where

$$C_i^-(n) = \max\left\{1 - S^{\mathrm{MAX}}, -nW_i^{\mathrm{MIN}}\right\}$$
$$C_i^+(n) = \min\left\{S^{\mathrm{MAX}} - 1, n\left(S^{\mathrm{MAX}} - W_i^{\mathrm{MIN}}\right)\right\}.$$

*Proof:*

$$\begin{aligned}\widetilde{w}_i(k,n) - w_i(k,n) &= \sum_{k}^{k+n-1} \widetilde{w}_{i,k} - \sum_{k}^{k+n-1} w_{i,k} \\ &= \sum_{k}^{k+n-1} \left(\widetilde{w}_{i,k}^{\mathrm{MIN}} + s_k\varphi_i\right) - \sum_{k}^{k+n-1} \left(w_{i,k}^{\mathrm{MIN}} + s_k\varphi_i\right) \\ &= \sum_{k}^{k+n-1} \widetilde{w}_{i,k}^{\mathrm{MIN}} - \sum_{k}^{k+n-1} w_{i,k}^{\mathrm{MIN}} \end{aligned} \quad (12)$$

where $w_{i,k}^{\mathrm{MIN}} = \min\{W_i^{\mathrm{MIN}}, q_{i,k}\}$ and $\widetilde{w}_{i,k}^{\mathrm{MIN}} = \min\{W_{i,k}^{\mathrm{MIN}}, q_{i,k}\}$.

Since queue $i$ remains backlogged during the entire interval of $n$ cycles, we can write $w_{i,k}^{\mathrm{MIN}} = W_i^{\mathrm{MIN}}$ and $\widetilde{w}_{i,k}^{\mathrm{MIN}} = W_{i,k}^{\mathrm{MIN}}$. Thus, (12) becomes

$$\begin{aligned}\widetilde{w}_i(k,n) - w_i(k,n) &= \sum_{k}^{k+n-1} W_{i,k}^{\mathrm{MAX}} - \sum_{k}^{k+n-1} W_i^{\mathrm{MAX}} \\ &= \sum_{k}^{k+n-1} W_{i,k}^{\mathrm{MAX}} - nW_i^{\mathrm{MAX}}. \end{aligned} \quad (13)$$

$$v_{i,k} = \begin{cases} v_{i,k-1}, & v_{i,k-1} \le 0 \text{ and } S_{i,k-1}^{\mathrm{HOL}} \le W_i^{\mathrm{MIN}} \quad (11a) \\ v_{i,k-1} + S_{i,k-1}^{\mathrm{HOL}} - W_i^{\mathrm{MIN}}, & v_{i,k-1} \le 0 \text{ and } S_{i,k-1}^{\mathrm{HOL}} > W_i^{\mathrm{MIN}} \quad (11b) \\ v_{i,k-1} - W_i^{\mathrm{MIN}}, & v_{i,k-1} > 0. \quad (11c) \end{cases}$$

From (9), we have

$$\begin{aligned} v_{i,k+1} &= v_{i,k} + W_{i,k}^{\mathrm{MAX}} - W_i^{\mathrm{MAX}} \\ v_{i,k+2} &= v_{i,k+1} + W_{i,k+1}^{\mathrm{MAX}} - W_i^{\mathrm{MAX}} \\ &\cdots \\ v_{i,k+n} &= v_{i,k+n-1} + W_{i,k+n-1}^{\mathrm{MAX}} - W_i^{\mathrm{MAX}}. \end{aligned}$$

Alternatively

$$v_{i,k+n} = v_{i,k} + \sum_{k}^{k+n-1} W_{i,k}^{\mathrm{MAX}} - nW_i^{\mathrm{MAX}}. \tag{14}$$

Substituting (14) into (13), we get

$$\widetilde{w}_i(k,n) - w_i(k,n) = v_{i,k+n} - v_{i,k}. \tag{15}$$

Equation (15) states that the difference between the actual and the optimal service during any interval of $n$ cycles is equal to the difference between the values of the overdraft counter at the beginning and at the end of this interval.

Using the bounds on the overdraft counter from Claim 2, we get

$$1 - S^{\mathrm{MAX}} \le v_{i,k+n} - v_{i,k} \le S^{\mathrm{MAX}} - 1. \tag{16}$$

Alternatively, we can expand

$$\begin{aligned} v_{i,k+n} - v_{i,k} = {} & (v_{i,k+n} - v_{i,k+n-1}) \\ & + (v_{i,k+n-1} - v_{i,k+n-2}) + \cdots + (v_{i,k+1} - v_{i,k}). \end{aligned}$$

Using the bounds from Claim 3, we get

$$-nW_i^{\mathrm{MIN}} \le v_{i,k+n} - v_{i,k} \le n(S^{\mathrm{MAX}} - W_i^{\mathrm{MIN}}). \tag{17}$$

Combining (16) and (17) and substituting the result into (15), we obtain upper and lower bounds on service (un)fairness as follows:

$$\begin{aligned} C_i^-(n) &= \max\left\{1 - S^{\mathrm{MAX}}, -nW_i^{\mathrm{MIN}}\right\} \\ C_i^+(n) &= \min\left\{S^{\mathrm{MAX}} - 1, n\left(S^{\mathrm{MAX}} - W_i^{\mathrm{MIN}}\right)\right\}. \end{aligned}$$

*Corollary 1:* Let queues $i$ and $j$ have the same guaranteed bandwidth ($W_i^{\mathrm{MIN}} = W_j^{\mathrm{MIN}}$) and the same weight ($\varphi_i = \varphi_j$). Then, the amount of service that queues $i$ and $j$ get in any $n$-cycle interval ($n = 1, 2, 3, \ldots$) in which they remain backlogged would differ by no more than $C_i^+(n) - C_i^-(n)$.

*Proof:* From $W_i^{\mathrm{MIN}} = W_j^{\mathrm{MIN}}$ and $\varphi_i = \varphi_j$, it follows that $w_i(k,n) = w_j(k,n)$. Thus

$$\begin{aligned} &\widetilde{w}_i(k,n) - \widetilde{w}_j(k,n) \\ &\qquad = (\widetilde{w}_i(k,n) - w_i(k,n)) - (\widetilde{w}_j(k,n) - w_j(k,n)). \end{aligned}$$

Substituting $\widetilde{w}_i(k,n) - w_i(k,n)$ and $\widetilde{w}_j(k,n) - w_j(k,n)$ by their bounds from Theorem 1, we get $C_i^-(n) - C_j^+(n) \le \widetilde{w}_i(k,n) - \widetilde{w}_j(k,n) \le C_i^+(n) - C_j^-(n)$.

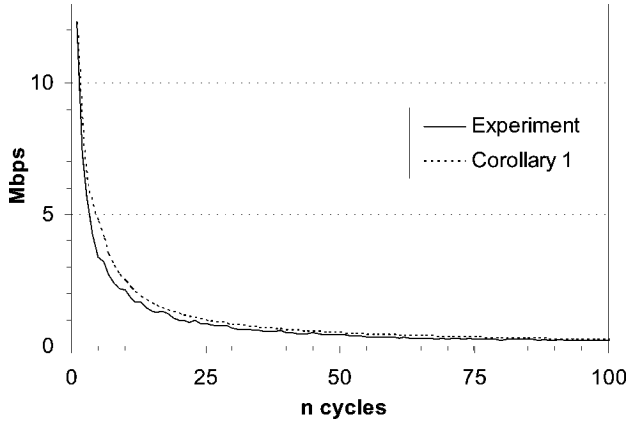



Fig. 15. Maximum difference in bandwidth allocated to queue 4 in ONU A and queue 4 in ONU B.

Since $W_i^{\mathrm{MIN}} = W_j^{\mathrm{MIN}}$, we have $C_i^-(n) = C_j^-(n)$ and $C_i^+(n) = C_j^+(n)$. Thus

$$|\widetilde{w}_i(k,n) - \widetilde{w}_j(k,n)| \le C_i^+(n) - C_i^-(n).$$

We illustrate Corollary 1 by an experiment in which we measure the difference in bandwidth allocated to two queues with the same guaranteed bandwidth and weight, and located in different ONUs. For example, we choose queue 4 in ONU A and queue 4 in ONU B in the experimental setup of Fig. 12. Fig. 15 presents two plots, the first being the maximum observed difference in bandwidths over the interval of 1000 s (measured as $|\widetilde{w}_i(k,n) - \widetilde{w}_j(k,n)|/nT$), and the second plot representing the maximum difference according to Corollary 1 [measured as $(C_i^+(n) - C_i^-(n))/nT$].

We observe that the difference in the allocated bandwidths quickly declines as the sampling window size $n$ increases. At 1-s sampling window ($n = 1000$), the maximum measured difference in allocated bandwidth was only $\sim 18$ kb/s. Results in Fig. 15 illustrate the short-term nature of loss of fairness due to HOL-blocking avoidance.

## VI. Conclusion

An important property of hierarchical resource-allocation algorithms is that the amount of information which each scheduling node processes is proportional only to number of children of this node, and does not depend on the number of consumers at the bottom of the hierarchy. On the other hand, to allow fairness among all the end consumers, the root scheduler should receive information from each consumer (especially if service consists of guaranteed and excess proportional-share parts as we have stated in Section II). This presents a fundamental conflict in a fair resource assignment in a remote-scheduling system; on the one hand, the algorithm should be hierarchical to achieve efficiency in the presence of large delays and limited control-plane bandwidth, and, on the other hand, there is a requirement to achieve resource-allocation fairness among the end consumers. In this paper, we have presented FQSE—a novel algorithm that successfully achieves both goals, it is hierarchical (each node knows only its immediate children) and it is cousin-fair.

```
CONSTRUCT_APPROX (E, ε)

    // first point in Ẽ corresponds to
    // first point in E
    Ẽ_1 = E_1
    n = 2

    for( k=2 to M-1 )
    {
       // find tangent to E (linear search)
       while( n<m-1 and TEST_TANGENT( Ẽ_{k-1}, n)>0 )
          n = n + 1
       TangentLine = ( Ẽ_{k-1} E_n)

       // find next intersection point
       // (linear search)
       while( n<m and TangentLine(X(E_n))-Y(E_n)<ε )
          n = n + 1

       // find segment parallel to (E_{n-1}E_n)
       // and shifted up by ε
       ShiftedSegment = (E_{n-1}E_n) + ε

       // notation A ⋂ B means point of
       // intersection of lines A and B
       <x', y'> = TangentLine ⋂ ShiftedSegment
       <x", y"> = TangentLine ⋂ (E_{m-1}E_m)

       // add k^th approximation point
       if( x' < x" )
          Ẽ_k = <x', y'>
       else
       {
          Ẽ_k = <x", y">
          Ẽ_{k+1} = E_m
          return SUCCESS
       }
    }

    return FAILURE
```

Fig. 16. CONSTRUCT_APPROX algorithm.

```
TEST_TANGENT( Point, n)

    if( SLOPE(E_{n-1}E_n) < SLOPE(PointE_n) )
       return -1
    else if( SLOPE(E_nE_{n+1}) > SLOPE(PointE_n) )
       return 1
    else
       return 0
```

Fig. 17. TEST_TANGENT function.

We have estimated FQSE's complexity and found that, at each node $i$, the scheduling work is proportional to the number of children $|D_i|$ of node $i$, and not to the total number of consumers as it would be in a flat-scheduling scheme.

In Section V, we proved cousin fairness property of FQSE and derived its fairness bounds. We showed that FQSE provides excellent fairness with a bound of less than one maximum-sized packet.

## APPENDIX

See Figs. 16–18.

```
FIND_APPROX( Δε )

    lower_ε = 0
    upper_ε = Δε

    // expanding search
    while( CONSTRUCT_APPROX( E, upper_ε ) == FAILURE )
    {
       lower_ε = upper_ε
       upper_ε = upper_ε × 2
    }

    // contracting search
    while(upper_ε - lower_ε > Δε )
    {
       ε = (lower_ε + upper_ε) / 2
       if( CONSTRUCT_APPROX( E, ε ) == FAILURE )
          lower_ε = ε
       else
          upper_ε = ε
    }

    CONSTRUCT_APPROX( E, upper_ε )
```

Fig. 18. FIND_APPROX procedure.

## ACKNOWLEDGMENT

The authors would like to thank C. Martel and D. Wiley of University of California, Davis for their advise on approximation algorithms.

## REFERENCES

[1] G. Kramer, B. Mukherjee, and A. Maislos, "Ethernet passive optical networks," in *IP Over WDM: Building the Next Generation Optical Internet*, S. Dixit, Ed. New York: Wiley, 2003.

[2] G. Kramer and G. Pesavento, "Ethernet passive optical network (EPON): Building a next-generation optical access network," *IEEE Commun.*, vol. 40, pp. 66–73, Feb. 2002.

[3] 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*, vol. 1, pp. 344–357, June 1993.

[4] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *J. Internetworking Res. Exp.*, pp. 3–26, Oct. 1990.

[5] J. C. R. Bennett and H. Zhang, "WF2Q: Worst-case fair weighted fair queueing," in *Proc. INFOCOM*, San Francisco, CA, Mar. 1996, pp. 120–128.

[6] L. Zhang, "Virtual clock: A new traffic control algorithm for packet switching networks," in *Proc. ACM SIGCOMM*, Sept. 1990, pp. 19–29.

[7] S. J. Golestani, "A self-clocked fair queueing scheme for broadband application," in *Proc. IEEE INFOCOM*, Toronto, ON, Canada, June 1994, pp. 636–646.

[8] P. Goyal, H. M. Vin, and H. Cheng, "Start-time fair queueing: A scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Trans. Networking*, vol. 5, pp. 690–704, Oct. 1997.

[9] D. Sala and A. Gummalla. PON functional requirements: Services and performance. presented at IEEE 802.3ah Meeting. [Online]. Available: http://grouper.ieee.org/groups/802/3/efm/public/jul01/presentations/sala_1_0701.pdf

[10] ITU-T Recommendation G.114, "One-Way Transmission Time," Transmission Systems and Media, Digital Systems and Networks, Telecommunication Standardization Sector of ITU, ser. G, May 2000.

[11] R. Bennet and H. Zhang, "Hierarchical packet fair queueing algorithms," *IEEE/ACM Trans. Networking*, vol. 5, pp. 675–689, Oct. 1997.

[12] C. Kalmanek, H. Kanakia, and S. Keshav, "Rate controlled servers for very high-speed networks," in *Proc. IEEE GLOBECOM*, San Diego, CA, Dec. 1990, pp. 1264–1273.

[13] J. L. Rexford, A. G. Greenberg, and F. G. Bonomi, "Hardware-efficient fair queueing architectures for high-speed networks," in *Proc. INFOCOM*, 1996, pp. 638–646.

[14] G. Kramer, B. Mukherjee, N. Singhal, A. Banerjee, S. Dixit, and Y. Ye, "Fair queueing with service envelopes (FQSE): A cousin-fair hierarchical scheduler and its application in Ethernet PON," Dept. Comput. Sci., Univ. California, Davis, CA, Tech. Rep. CSE-2003–6, 2003.
[15] G. Kramer, B. Mukherjee, and G. Pesavento, "Ethernet PON (ePON): Design and analysis of an optical access network," *Photonic Network Commun.*, vol. 3, no. 3, pp. 307–319, July 2001.
[16] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round-robin," *IEEE/ACM Trans. Networking*, vol. 4, pp. 375–385, June 1996.
[17] IEEE P802.3ah. (2003) "Ethernet in the first mile (EFM)" Task Force Objectives. [Online]. Available: http://grouper.ieee.org/groups/802/3/efm/public/jan03/objectives_1_0103.pdf
[18] M. S. Taqqu, W. Willinger, and R. Sherman, "Proof of a fundamental result in self-similar traffic modeling," *ACM/SIGCOMM Comput. Commun. Rev.*, vol. 27, pp. 5–23, 1997.

**Glen Kramer** (S'00–M'03) received the B.S. degree in computer engineering from Kishinev Polytechnic Institute, Moldova, in 1992, and the M.S. and Ph.D. degrees in computer science from the University of California, Davis, in 2000 and 2003, respectively.

From May 1999 to December 2002, he worked in the Advanced Technology Laboratory, Alloptic, Inc., where he was involved in the design and analysis of EPON scheduling protocols, investigated bandwidth utilization issues in access networks, and authored several patents. Currently, he is with Teknovus, Inc., Petaluma, CA, as a System Architect. He is a Founder of EPON Forum and EPON protocol clause editor in IEEE 802.3ah "Ethernet in the First Mile" task force. He is also a Research Associate with the Networks Research Laboratory, UC Davis. His main areas of research are traffic scheduling, QoS and flow prioritization, network traffic characterization, and effects of traffic shaping.

**Amitabha Banerjee** (S'02) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Delhi, India, in May 2000, and the M.S. degree in computer science from the University of California, Davis, in March 2004. He is currently working toward the Ph.D. degree in the Computer Science Department, University of California, where he is a Research Assistant in the Networks Research Laboratory.

From June 2000 to July 2002, he was with Tavant Technologies, Santa Clara, CA. His research interests include quality-of-service (QoS) and fairness issues in access networks.

**Narendra K. Singhal** (S'00) received the B.Tech. degree (Honors) in computer science and engineering from the Indian Institute of Technology, Kharagpur, India, in 1998 and the M.S. degree in computer science from the University of California, Davis, in 2000. He is currently working toward the Ph.D. degree at the University of California.

From June 1998 to September 1999, he was with Verifone India, Ltd., a fully owned subsidiary of Hewlett Packard. His research interests include optical network design, multicasting, survivability, and network optimization.

Mr. Singhal received the Professors for the Future Fellow (PFTF) 2002–2003 Award from the University of California.

**Biswanath Mukherjee** (S'82–M'87) received the B.Tech. (Honors) degree from the Indian Institute of Technology, Kharagpur, India, in 1980 and the Ph.D. degree from the University of Washington, Seattle, in June 1987, where he held a GTE Teaching Fellowship and a General Electric Foundation Fellowship.

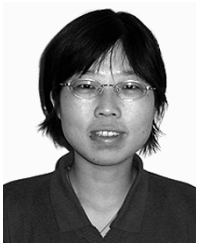
In July 1987, he joined the University of California, Davis (UC Davis), where he has been a Professor of Computer Science since July 1995, and served as Chairman of Computer Science from September 1997 to June 2000. He is the author of *Optical Communication Networks* (New York: McGraw-Hill, 1997), a book which received the Association of American Publishers, Inc.'s 1997 Honorable Mention in Computer Science. He is a Member of the Board of Directors of IPLocks, Inc., a Silicon Valley startup company. He has consulted for and served on the Technical Advisory Board of a number of startup companies in optical networking. His research interests include lightwave networks, network security, and wireless networks.

Dr. Mukherjee is co-winner of Paper Awards presented at the 1991 and the 1994 National Computer Security Conferences. He serves or has served on the Editorial Boards of the IEEE/ACM TRANSACTIONS ON NETWORKING, the *IEEE Network*, ACM/Baltzer *Wireless Information Networks* (WINET), *Journal of High-Speed Networks, Photonic Network Communications*, and *Optical Network Magazine*. He also served as Editor-at-Large for optical networking and communications for the IEEE Communications Society. He served as the Technical Program Chair of the IEEE INFOCOM'96 Conference. He is a winner of the 2004 Distinguished Graduate Mentoring Award at UC Davis. Two Ph.D. dissertations (by L. Sahasrabuddhe and K. Zhu), which were supervised by Dr. Mukherjee, were winners of the 2000 and 2004 UC Davis College of Engineering Distinguished Dissertation Awards.

**Sudhir Dixit** (S'75–A'80–M'80–SM'94) received the B.E. degree from Maulana Azad National Institute of Technology (MANIT), Bhopal, India, the M.E. degree from Birla Institute of Technology and Science (BITS), Pilani, India, and the Ph.D. degree from the University of Strathclyde, Glasgow, Scotland, all in electrical engineering. He also received the M.B.A. degree from the Florida Institute of Technology, Melbourne, FL.

He is a Research Fellow at Nokia Research Center, Burlington, MA, and concentrates on research in next-generation wireless networks. Prior to this, he was a Senior Research Manager from 1996 to October 2003 and managed the Mobile Internet Performance Group and its earlier incarnations, specializing in pervasive communications, content networking and optical networking. From 1991 to 1996, he was a Broadband Network Architect at NYNEX Science and Technology, New York, (now Verizon Communications). Prior to that he held various engineering and management positions at other major companies, e.g., GTE, Motorola, Wang, Harris, and STL (now Nortel Europe Labs). He has published or presented over 150 papers and has 30 patents either granted or pending. He has coedited a book *Wireless IP and Building the Wireless Internet* (Norwood, MA: Artech House, 2002), edited a book on *IP Over WDM* (New York: Wiley, 2003), and a third book on *Content Networking in the Mobile Internet* (New York: Wiley, 2004).

Dr. Dixit has been a Technical Co-Chair and a General Chair of the IEEE International Conference on Computer, Communications and Networks, in 1999 and 2000, respectively, a Technical Co-Chair of the SPIE Conference Terabit Optical Networking in 1999, 2000, and 2001, respectively, a General Chair of the Broadband Networking in the Next Millennium Conference in 2001, a General Co-Chair of the OptiComm 2002 Conference, and a General Chair of International Conference on Communication and Broadband Networking both in 2003 (Bangalore, India) and in 2004 (Kobe, Japan). He has also been an ATM Forum Ambassador since 1996. He has served as a Guest Editor several times for the *IEEE Network*, the *IEEE Communications Magazine*, ACM MONET, and SPIE/Kluwer *Optical Networks Magazine*. Currently, he is on the Editorial Boards of the *Wireless Personal Communications Journal*, *Journal of Communications and Networks*, *IEEE Communications Magazine Optical Supplement*, and the *International Journal on Wireless and Optical Communications*.



**Yinghua Ye** (S'98–M'00) received the Ph.D. degree in electrical engineering from City University of New York (CUNY), in 2000.

She joined Nokia Research Center, Burlington, MA, in June 2000. Currently, she is a Senior Research Engineer in the Pervasive Computing Group, Nokia Research Center. From July 2003, she has been working on universal plug and play (UPnP) related project. She has published more than 35 papers in conferences and journals, and currently has two U.S. patents pending and one provisional application in the field of optical networking. Her research interests include security in mobile ad hoc networks, service discovery, architecture design, network survivability, traffic engineering, and real-time provisioning in optical networks.

Dr. Ye has served as Technical Committee Member for Opticom 2002 and OptimCom 2003. She was actively involved with IEEE802.3 Standardization activities in 2002 and made some contributions to MPCP.