# Interleaved Polling with Adaptive Cycle Time (IPACT): A Dynamic Bandwidth Distribution Scheme in an Optical Access Network.

Glen Kramer
Department of Computer Science
University of California, Davis, CA 95616, USA
Tel: 1.925.245.7645; Fax: 1.925.245.7601
E-mail: *kramer@cs.ucdavis.edu*

Biswanath Mukherjee
Department of Computer Science
University of California, Davis, CA 95616, USA
Tel: 1.530.752.5129; Fax: 1.530.752.4767
E-mail: *mukherjee@cs.ucdavis.edu*

Gerry Pesavento
Advanced Technology Lab.
Alloptic, Inc.
Livermore, CA 94550, USA
Tel: 1.925.245.7647; Fax: 1.925.245.7601
E-mail: *gerry.pesavento@alloptic.com*

July 2001

**Abstract**

While in recent years backbone bandwidth has experienced substantial growth, little has changed in the access network. "Last mile" still remains the bottleneck between a high capacity LAN or home network and the backbone. Passive Optical Network (PON) is a technology viewed by many as an attractive solution to this problem.

In this study, we discuss and evaluate design issues for PON access networks. Specifically, to drive the cost of an access network down, it is very important to have an efficient, scalable solution. We believe that a PON based on polling, with data encapsulated in Ethernet frames, possesses the best qualities, such as dynamic bandwidth distribution, use of a single downstream and a single upstream wavelength, ability to provision a fractional wavelength capacity to each user, and ease of adding a new user.

To support dynamic bandwidth distribution, we propose an interleaved polling algorithm. We then suggest a scheme for in-band signaling that allows using a single wavelength for both downstream data and control message transmission.

To obtain realistic simulation results, we generated synthetic traffic that exhibits the properties of self-similarity and long-range dependence. We then analyzed the network performance and its effect on various types of traffic, e.g., best-effort data traffic, VBR video traffic and CBR streams.

**Keywords:** access network, local loop, passive optical network, PON, interleaved polling, self-similar traffic, long-range dependence, dynamic bandwidth distribution.

## 1 Introduction

While in recent years backbone bandwidth has experienced substantial growth, little has changed in the access network. "Last mile" still remains the bottleneck between a high capacity LAN or home network and a backbone. Digital Subscriber Line (DSL) and Cable Modem (CM) technologies offer some improvement, but still do not provide enough bandwidth for emerging services such as Video-On-Demand (VoD) or two-way video conferencing. Also, not all end-users can be covered by those technologies due to distance limitations.

Passive Optical Network (PON) is a technology viewed by many as an attractive solution to this problem [1, 2]. A PON is a point-to-multipoint optical network with no active elements in

the signals' path from source to destination. The only interior elements used in PON are passive combiners, couplers, and splitters.

Advantages of using a PON for local access networks are numerous:

- A PON allows for longer distances between central offices and customer premises. While with Digital Subscriber Line (DSL) the maximum distance between the central office and the customer is only 18000 feet (approximately 5.5 km), a PON local loop can operate at distances of over 20 km.
- A PON minimizes fiber deployment in both the local exchange and the local loop.
- A PON provides higher bandwidth due to deeper fiber penetration. While the fiber-to-the-building (FTTB), fiber-to-the-home (FTTH), or even fiber-to-the-PC (FTTPC) solutions have the ultimate goal of fiber reaching all the way to customer premises, fiber-to-the-curb (FTTC) may be the most economical deployment today.
- As a point-to-multipoint network, a PON allows for downstream video broadcasting.
- A PON eliminates the necessity of installing multiplexers and demultiplexers in the splitting locations, thus relieving network operators from the gruesome task of maintaining them and providing power to them. Instead of active devices in these locations, a PON has passive components that can be buried into the ground at the time of deployment.
- A PON allows easy upgrades to higher bit rates or additional wavelengths.

A cost analysis presented in [3] shows that, in many situations, deploying fiber is now less costly than deploying copper. On the other hand, because an access network aggregates traffic from a relatively small number of subscribers (compared to metro or regional networks), it is very cost sensitive. Therefore, a PON design should not require over-provisioning and should allow for incremental deployment.

In [4], we discussed the advantages of using time-division multiple access (TDMA) in a PON, namely, the scalability and ability to provide a fraction of a wavelength capacity to a user, single wavelength for all upstream channels, and a single receiver in the head end, etc. However, we also showed that a considerable amount of bandwidth was wasted due to timeslots not being filled to capacity. To make the cost of a PON-based access network lower, it is very important to utilize bandwidth efficiently.

In Section 4, we propose a media access scheme based on polling. The considered architecture (Section 2) uses a polling scheme to deliver data encapsulated in Ethernet packets from a collection of Optical Network Units (ONUs) to a central Optical Line Terminal (OLT) over the PON access network. The OLT, in turn, is connected to the rest of the Internet.

To avoid the accumulation of walk times (switchover times) associated with polling, we employ an *interleaved* scheme where multiple polling requests are overlapped in time. We then discuss an efficient way to use in-band control signaling to perform the polling.

A simulation model described in Section 2 is used to analyze the system's performance, such as bounds on packets delay, queue occupancy, and packet loss probability. To obtain an accurate and realistic performance analysis, it is very important to simulate system behavior with appropriate traffic injected into the system. A simulation analysis was performed using synthetic traffic traces that exhibit the property of *self-similarity* and *long-range dependence* (*LRD*) [5]. In Section 2.3, we give a brief overview of real traffic characteristics and discuss a method we used to generate the synthetic traffic.

In Section 4.5, we discuss constituent parts of packet delay and consider various schemes for ensuring fairness, viz., preventing bandwidth monopolization by a single source or a group of sources. We then present the simulation results in Section 5.

Section 6 concludes this study and suggests the future research directions.

## 2 Design of an Access Network Based on PON Technology

Figure 1 shows several topologies suitable for the access network: (a) tree, (b) ring, or (c) bus. A PON can also be deployed in a redundant configuration as a double ring or a double tree.

All transmissions in a PON are performed between Optical Line Terminal (OLT) and Optical Network Units (ONUs). Therefore, in the downstream direction (from OLT to ONUs), a PON is a point-to-multipoint network, and in the upstream direction it is a multipoint-to-point network.

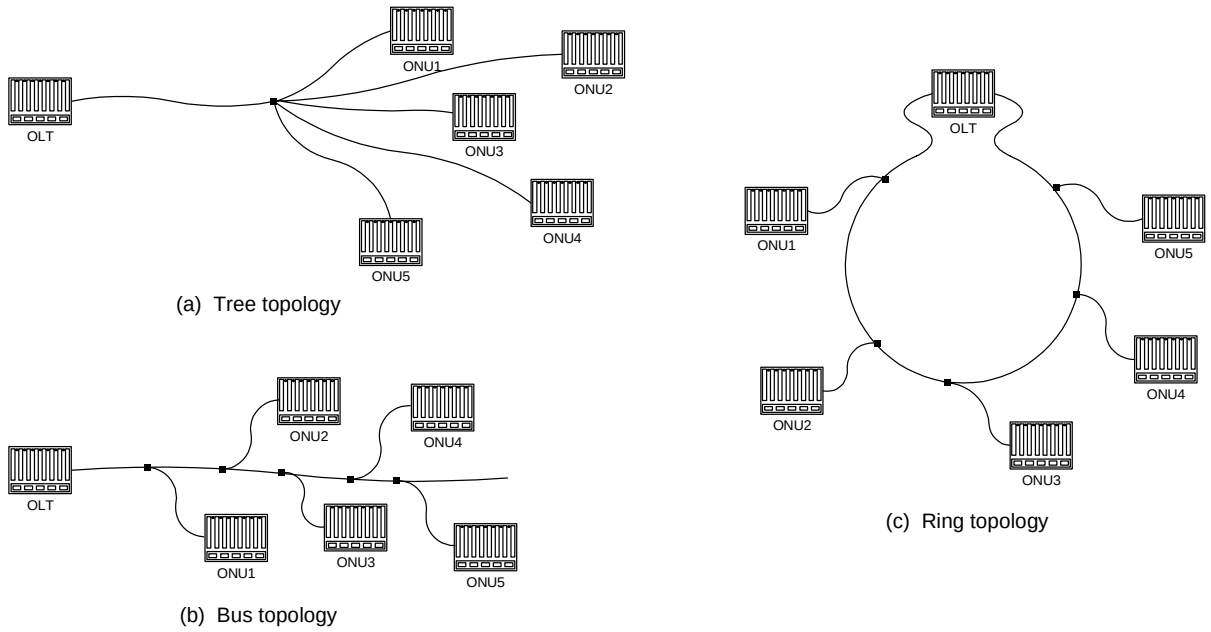


**Figure 1. Topologies used in Passive Optical Networks.**

The OLT resides in the local exchange (central office), connecting the optical access network to an IP, ATM, or SONET backbone. The ONU is located either at the curb (FTTC solution), or at the end-user location (FTTH, FTTB solutions), and provides broadband voice, data, and video services.

## 2.1 Which Layer 2?

The Full Service Access Network (FSAN) standard defines an optical access network that uses ATM as its layer 2 protocol. In 1995, when the FSAN initiative started, ATM had high hopes of becoming the prevalent technology in LAN, MAN, and backbone. However, since that time, Ethernet technology has leapfrogged. Ethernet has become a universally accepted standard, with over 320 million port deployments worldwide [6]. High-speed Gigabit Ethernet deployment is widely accelerating and 10 Gigabit Ethernet products are in final stages of development. Ethernet, which is easy to scale and manage, is winning new ground in MAN and WAN. Considering the fact that about 95% of LANs use Ethernet, it becomes clear that ATM PON is not the best choice to interconnect two Ethernet networks.

One of ATM's shortcomings is the fact that a dropped or corrupted cell will invalidate the entire IP datagram. However, the remaining cells carrying the portions of the same IP

datagram will propagate further, thus consuming network resources unnecessarily. Also, ATM imposes an overhead (commonly referred to as "cell tax") on variable-length IP packets. For example, for the tri-modal packet size distribution reported in [7], the cell tax is approximately 14%, i.e., to send the same amount of user's data an ATM network must transmit 14% more bytes than an Ethernet network (counting 64-bit preamble in Ethernet and 12 bytes of overhead associated with AAL-5 in ATM). And finally, ATM equipment is significantly more expensive than Ethernet [6].

All of the above makes FSAN's future, relying on ATM transport, rather grim. Ethernet PONs, on the contrary, look like a preferred choice. Newly-adopted Quality of Service (QoS) techniques have made Ethernet networks capable of support voice, data, and video. Ethernet is an inexpensive technology which is interoperable with a variety of legacy equipment. In this study we will focus on Ethernet PONs.

## 2.2 Model Description

In this study, we consider an access network consisting of an OLT and $N$ ONUs connected using a passive optical network (Figure 2). Every ONU is assigned a downstream propagation delay (from the OLT to the ONU) and an upstream propagation delay (from the ONU to the OLT.) While with a tree topology both downstream and upstream delays are the same, with a ring topology delays will be different. To keep the model general we assume independent delays and select them randomly (uniformly) over the interval [50 μs, 100 μs]. These values correspond to distances between the OLT and ONUs ranging from 10 to 20 km.

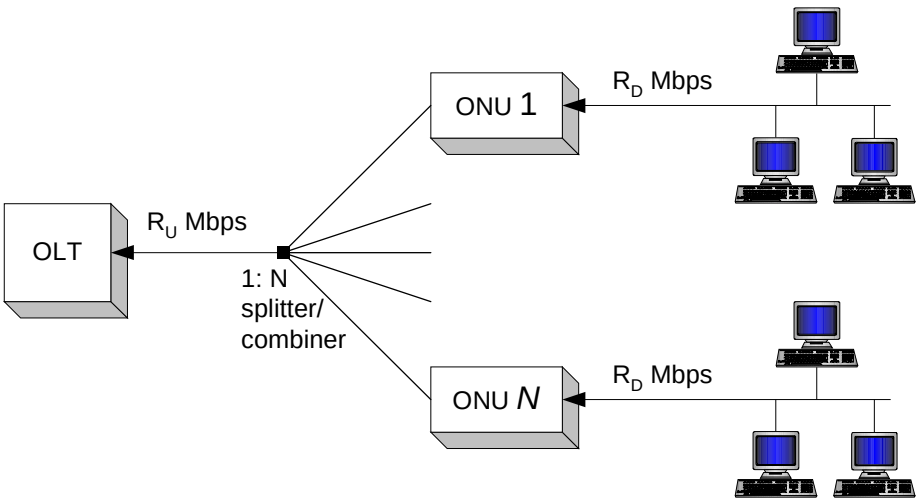


**Figure 2. Access network based on PON.**

Because Ethernet is broadcasting by nature, in the downstream direction (from network to user), it fits perfectly with the Ethernet PON architecture: packets are broadcast by the OLT and extracted by their destination ONU based on the media-access control (MAC) address.

In the upstream direction (from user to network), the ONUs should share the channel capacity and resources. The reader is referred to [4] for a discussion of various multiple access schemes and their applicability to PON-based access networks. We believe the TDMA approach is a preferential method of channel sharing in an access network as it allows using a single upstream wavelength and results in a very cost-effective solution. However, in [4] we also showed the limitation of fixed TDMA approach, viz., the lack of statistical multiplexing.

From the access side, traffic may arrive at an ONU from a single user or from a gateway of a local-area network (LAN), i.e., traffic may be aggregated from a number of users. Packets should be buffered in the ONU until the ONU is allowed to transmit the packets. The transmission speed of the PON and the user access link may not necessarily be the same. In our model, we consider $R_D$ Mbps to be the data rate of the access link from a user to an ONU, and $R_U$ Mbps to be the rate of the upstream link from an ONU to the OLT (see Figure 2). We should mention here that, if $R_U \geq N \times R_D$, then the bandwidth utilization problem does not exist, as the system throughput is higher than the peak aggregated load from all ONUs. In this study, we consider a system with $N$=16 and $R_D$ and $R_U$ being 100 Mbps and 1000 Mbps, respectively.

## 2.3 Synthetic Traffic Generation

To obtain an accurate and realistic performance analysis, it is important to simulate the system behavior with appropriate traffic injected into the system. In this section, we will give a brief overview of real traffic characteristics and discuss the method used to generate the synthetic traffic.

There is an extensive study showing that most network traffic flows (i.e., generated by http, ftp, variable-bit-rate (VBR) video applications, etc.) can be characterized by self-similarity and long-range dependence (LRD) (see [8] for an extensive reference list).

Self-similarity of a (weakly) stationary process *X(t)* is a characteristic of the scaling behavior of its auto-covariance function $\gamma(k)$:

$$\gamma(k) = E[(X(t) - \mu)(X(t+k) - \mu)] \quad (1)$$

Let $\gamma^{(m)}(k)$ be the auto-covariance function of a process $X^{(m)}(t)$, where

$$X^{(m)}(t) = \frac{1}{m}(X(mt - m + 1) + \ldots + X(mt)) \quad (2)$$

i.e., $X^{(m)}(t)$ is a *m*-times aggregated process $X(t)$. Then, the process $X(t)$ is exactly self-similar if $\gamma^{(m)}(k) = \gamma(k)$ and asymptotically self-similar if $\gamma^{(m)}(k) \to \gamma(k)$, as $m \to \infty$. The measure of a process's self-similarity is a Hurst parameter H (½ < H < 1). The self-similarity can be viewed as an ability of an aggregated process to "preserve" the burstiness of the original process, viz., the property of slowly decaying variance:

$$\text{var}(X^{(m)}) \sim m^{2H-2} \quad (3)$$

In the context of network traffic, this means that aggregating traffic over large time intervals reduces the burstiness very slowly (compared to non-self-similar traffic).

The property of long-range dependence refers to a non-summable auto-correlation function $\rho(k) = \gamma(k)/\sigma^2$:

$$\sum_{k=-\infty}^{\infty} \rho(k) = \infty \quad (4)$$

The long-range dependence results from a heavy-tailed distribution of the corresponding stochastic process. Heavy-tailedness refers to the rate of tail decay of the complementary distribution function. In a heavy-tail distribution, the decay obeys power law:

$$P[X > x] \sim cx^{-\alpha}, \quad \text{as } x \to \infty \quad \text{and} \quad 1 < \alpha < 2 \tag{5}$$

As a result, the probability of an extremely large observation in LRD process is non-negligible. In the context of network traffic, this means that extremely large bursts of data (packet trains) and extremely long periods of silence (inter-arrival times) will occur from time to time. This is one of the reasons why analytic models employing traditional negative exponential distribution often provide overly optimistic estimates for the delays and queue sizes – the probability of an extreme event is negligible. We refer the reader to [9] and [10] for a more rigorous treatment of self-similarity and long-range dependence.

To generate self-similar traffic, we used the method described in [11], where the resulting traffic is an aggregation of multiple streams, each consisting of alternating Pareto-distributed ON/OFF periods.

Pareto distribution is a heavy-tailed distribution with the probability density function

$$f(x) = \frac{\alpha b^{\alpha}}{x^{\alpha+1}}, \quad x \geq b, \tag{6}$$

where $\alpha$ is a shape parameter ($1 < \alpha < 2$), and $b$ is a location parameter.

Pareto distribution with $1 < \alpha < 2$ has a finite mean and an infinite variance.

In our implementation every stream generates Ethernet packets that are transmitted in packet trains (bursts). The number of packets per burst (ON period) follows the Pareto distribution with a minimum of 1 (i.e., the smallest burst consists of only 1 packet) and shape parameter $\alpha = 1.4$. The choice of $\alpha$ was prompted by measurements on actual Ethernet traffic performed by Leland et al. [5]. They reported the measured Hurst parameter of 0.8 for moderate network load. The relationship between the Hurst parameter and the shape parameter $\alpha$ is $H = (3 - \alpha)/2$ (see [11]). Thus, $\alpha = 1.4$ should result in $H$=0.8.

OFF periods (intervals between the packet trains) also follow the Pareto distribution, though with the shape parameter $\alpha = 1.2$. We used heavier tail for the distribution of the OFF periods because the OFF periods represent a stable state in a network, i.e., a network can be in OFF state (no packet transmission) for an unlimitedly long time, while the durations of the ON

periods are ultimately limited by network resources and (necessarily finite) file sizes. The location parameter *b* for the OFF periods was chosen such as to obtain a desired load $\tilde{\phi}_i$ from the given source *i*:

$$\tilde{\phi}_i = \frac{E[ON_i]}{E[ON_i] + E[OFF_i]} \tag{7}$$

where $E[ON_i]$ and $E[OFF_i]$ are expected lengths (durations) of ON and OFF periods of source *i*.

Traffic in every ONU was generated by aggregating $n$=32 sources. We experimented with various numbers of sources and found that the burstiness of the traffic (Hurst parameter) does not change with *n* if the total load is fixed, i.e., if $\tilde{\phi}_i \sim 1/n$ .

Figure 3 shows a variance-time log-log plot we used to verify the correctness of our traffic generator. We ran the generator with $n$ = 32, 128, 512, and 2048 sources. The plot shows linear dependency (except tail region) with slope $s$ = –0.4. From equation (3) we expect the log-log plot to have a slope $s = 2H - 2$. That results in $H = 1 + s/2 = 0.8$, as expected.

Note that starting with *log(m)* = 3.3 the variance decay increases. There are two reasons for that. The first reason is the fact that synthetically generated traces (as well as those collected on real networks) have a truncated tail, i.e., after some threshold the tail decays exponentially or faster. In our generator, the tail of the distribution function is truncated at the value $2^{32}$-1, i.e., the maximum burst size in our generator is $2^{32}$-1 packets. However, the second reason is more important, viz., the fact that simulations (as well as real network measurements) are performed with a finite set of observations. To build the Variance-Time plot in Figure 3, we generated 15 million packets. The maximum observed burst (length of ON period) was in the order of 100,000 packets long. Starting with the aggregation level of $10^{3.3}$ ms ≈ 2.4 seconds, the number of ON periods that span multiple intervals of that size started to decrease rapidly. In other words, the number of ON (and OFF) periods per interval started to increase proportionally to the increase of the interval size itself. That resulted (by the law of large numbers) in more pronounced averaging, and thus, in faster variance decay. Even though it is an artifact here, it does not affect the simulation results - the intervals (and, therefore, the delays) in the range of seconds are not of practical interest, as averaging of traffic on such a scale is beyond buffering capacities or allowable delays.

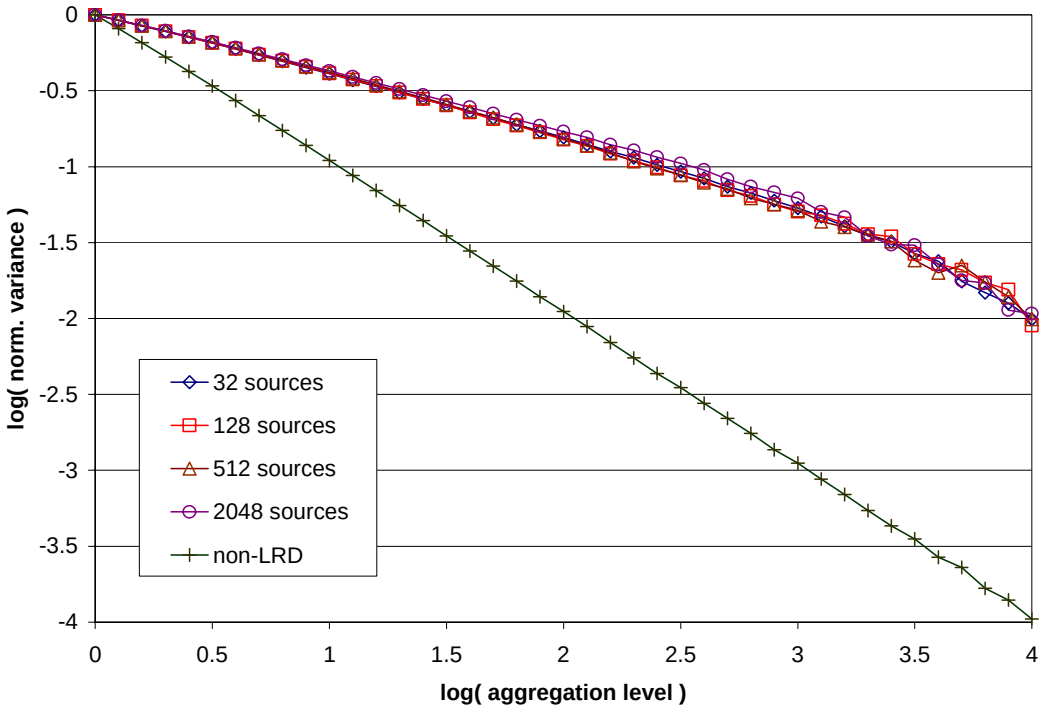


**Figure 3. Variance-Time log-log plot.**

The fact that the plots for all $n$ ($n$ – number of ON/OFF sources) coincide motivated our decision to use $n = 32$. The last plot called "*non-LRD*" was obtained on the same generator, but with exponentially-distributed ON/OFF periods. This distribution possesses no long-range dependence and its variance-time log-log plot is expected to have a slope of –1 ( $\mathrm{var}(X^{(m)}) \sim m^{-1}$ ). We plotted it just to verify that our variance normalizations and thus, slopes, are correct.

Figure 4 illustrates the way the traffic was generated in an individual ONU. Within the ON period, every source generates packets back to back (with a small fixed interval in between). Every source assigns a specific priority value to all its packets. Packets generated by $n$ sources are aggregated (multiplexed) on a single line such that packets from different sources do not overlap. We used the FCFS discipline to multiplex the packets. After that the packets are forwarded to the respective queues based on their priority assignments and the queues are served in order of their priorities.

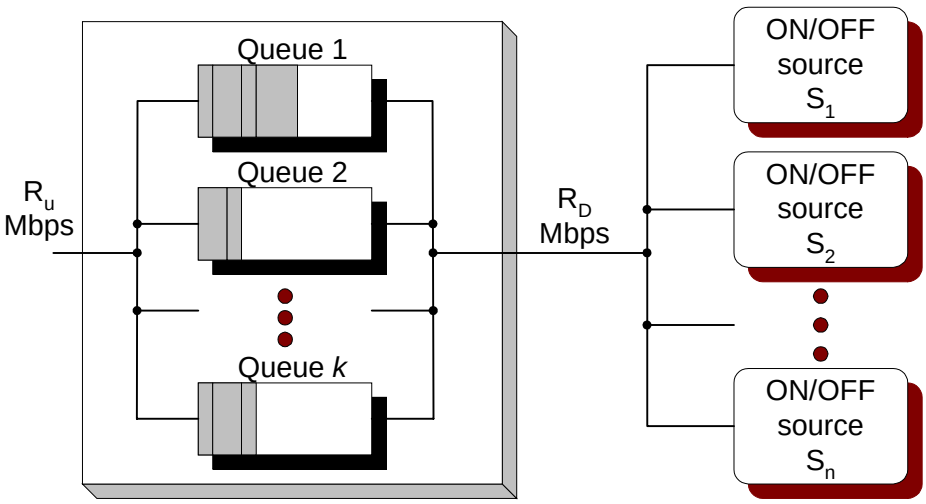


**Figure 4. Traffic generation in the ONU.**

Load aggregated from all $n$ sources in an ONU is called *offered ONU load* (OOL) and denoted by $\phi$:

$$\phi = \sum_{i=1}^{n} \tilde{\phi}_i \tag{8}$$

*Offered network load* (ONL) $\Phi$ is the sum of the loads offered by each ONU and scaled based on $R_D$ and $R_U$ rates. Clearly, since the network throughput is less than the aggregated peak bandwidth from all ONUs, the ONL can exceed 1:

$$\Phi = \frac{R_D}{R_U} \sum_{j=1}^{N} \phi^{[j]} \tag{9}$$

It is important to differentiate between offered load and *effective load*. The effective ONU load (EOL) is denoted $\varpi$ and results from the data (packets) that have been sent out by the ONUs. Thus, the EOL is equal to the OOL only if the packet loss rate is zero. In general $\varpi \le \phi$.

The EOL generated by the ONU $j$ is denoted $\varpi^{[j]}$. *Effective network load* (ENL) $\Omega$ is just a sum of the EOLs generated by all ONUs with a corresponding scaling coefficient based on the PON and user link bit rates:

$$\Omega = \frac{R_D}{R_U} \sum_{j=1}^{N} \varpi^{[j]} \tag{10}$$

Every ONU may have $k$ queues, $k$ = 1,2,3... which are served in order of their priority (queue $u$ has higher priority than queue $v$ if $u > v$). Every ONU has a finite buffer of size

Q. The memory is allocated to different queues based on demand and priority, i.e., if the entire buffer is occupied and a packet with priority *k* arrives, the lowest-priority non-empty queue will drop one or more packets, so that the higher-priority queue *k* can store the new packet. In our simulations, buffer size Q was set to 10 Mbyte.

### 2.4 Bandwidth Utilization Problem

In our previous study [4], we considered a TDMA scheme, in which every ONU gets a fixed timeslot. While this scheme is very simple, it had a drawback that no statistical multiplexing between the ONUs was possible.

In has been shown that network traffic exhibits a high degree of burstiness [5]. Traffic aggregation does not solve the problem as the variance of aggregated traffic decreases much slower that the variance of a conventional Poisson process (see Section 2.3). The long-range dependence (heavy-tailedness) of the traffic results in a situation where some timeslots overflow even under very light load resulting in packets being delayed for several timeslot periods. It is also true that some timeslots remain underutilized (not filled completely) even if the traffic load is very high. That leads to the PON bandwidth being underutilized.

A dynamic scheme that reduces the timeslot size when there is no data would allow the excess bandwidth to be used by other ONUs. The challenge of implementing such a scheme is in the fact that the OLT does not know exactly how many bytes of data each ONU has. The observed burstiness of traffic at all timescales should convince the reader that trying to predict the timeslot occupancy based on prior history will not work. If the OLT is to make an accurate timeslot assignment, it should know exactly how many bytes are waiting in a given ONU.

One approach could be to relieve the OLT from timeslot assignment altogether and leave it to ONUs. This scheme is somewhat similar to a token ring, except that it is a passive ring. In such a scheme, every ONU, before sending its data, will send a special message announcing how many bytes it is about to send. The ONU that is scheduled next (say, in round-robin fashion) will monitor the transmission of the previous ONU and will time its transmission such that it arrives to the OLT right after the transmission from the previous ONU. Thus, there will be no collision and no bandwidth will be wasted. This scheme is similar to roll-call polling.

However, this scheme has a major limitation: it requires connectivity (communicability) between ONUs. That imposes some constraints on PON topology, namely, the network should

be deployed as a ring or as a broadcasting star. This requirement is not always satisfiable as (a) it may require more fiber to be deployed, or (b) fiber plant with different topology might be already pre-deployed. In general, we want our algorithm to be able to support whatever PON topology is given.

In an access network, we can count only on connectivity from the OLT to every ONU and every ONU to the OLT. That is true for all PON topologies. Therefore, the OLT is the only device that can arbitrate the time-divided access to the shared channel.

In Section 3, we propose an OLT-based polling scheme, similar to hub polling (see, for example [12]). Simple hub polling, however, has a disadvantage of increasing the polling cycle due to accumulation of walk times (switchover times). In access networks, this problem is even more pronounced than in LANs, as the propagation delays in the access network are much higher. For example, in a PON with 16 ONUs each having 200 μs round-trip delay (ONU and OLT are 20 km apart) the accumulated walk time is 3.2 ms. This is how much time would be wasted in every cycle in the hub polling. Our proposed algorithm uses an interleaved polling approach where the next ONU is polled before the transmission from the previous one has arrived. This scheme provides statistical multiplexing for ONUs and results in efficient upstream channel utilization.

## 3 Interleaved Polling Algorithm

In this section, we give a high-level overview of the proposed algorithm. For simplicity of illustration, we will consider a system with only three ONUs.

1. Let us imagine that at some moment of time $t_0$ the OLT knows exactly how many bytes are waiting in each ONU's buffer and the Round-Trip Time (RTT) to each ONU. The OLT keeps this data in a polling table shown in Figure 5.a. At time $t_0$ , the OLT sends a control message to ONU1, allowing it to send 6000 bytes (see Figure 5.a). We will call such a message a *Grant.* Since, in the downstream direction, the OLT broadcasts data to all ONUs, the Grant should contain the ID of the destination ONU, as well as the size of the granted window (in bytes).
2. Upon receiving the Grant from the OLT, ONU1 starts sending its data up to the size of the granted window (Figure 5.b). In our example – up to 6000 bytes. At the same time, the ONU keeps receiving new data packets from its user. At the end of its transmission window, ONU1 will generate its own control message (*Request*). The Request sent by

ONU1 tells the OLT how many bytes were in ONU1's buffer at the moment when the Request was generated. In our case there were 550 bytes.

3. Even before the OLT received a reply from ONU1, it knows when the last bit of ONU1's transmission will arrive. This is how the OLT calculates this:
   (a) the first bit will arrive exactly after the RTT time. The RTT in our calculation includes the actual round-trip time, Grant processing time, Request generating time, and a preamble for the OLT to perform bit- and byte-alignment on received data, i.e., it is exactly the time interval between sending a Grant to an ONU and receiving data from the same ONU.
   (b) since the OLT knows how many bytes (bits) it has authorized ONU1 to send, it knows when the last bit from ONU1 will arrive. Then, knowing RTT for ONU2, the OLT can schedule a Grant to ONU2 such that first bit from ONU2 will arrive with a small guard interval after the last bit from ONU1 (Figure 5.b). The guard intervals provide protection for fluctuations of round-trip time and control message processing time of various ONUs. Additionally, the OLT receiver needs some time to readjust its sensitivity due to the fact that signals from different ONUs may have different power levels because ONUs are located at different distances from the OLT (far-near problem).

4. After some time, the data from ONU1 arrives. At the end of the transmission from ONU1, there is a new Request that contains information of how many bytes were in ONU1's buffer just prior to the Request transmission. The OLT will use this information to update its polling table (see Figure 5.c).

   By keeping track of times when Grants are sent out and data is received, the OLT constantly updates the RTT entries for the corresponding ONUs

5. Similarly to Step 4, the OLT can calculate the time when the last bit from ONU2 will arrive. Hence, it will know when to send the Grant to ONU3 so that its data is tailed to the end of ONU2's data. After some more time, the data from ONU2 will arrive. The OLT will again update its table, this time the entry for the ONU2 (see Figure 5.d).

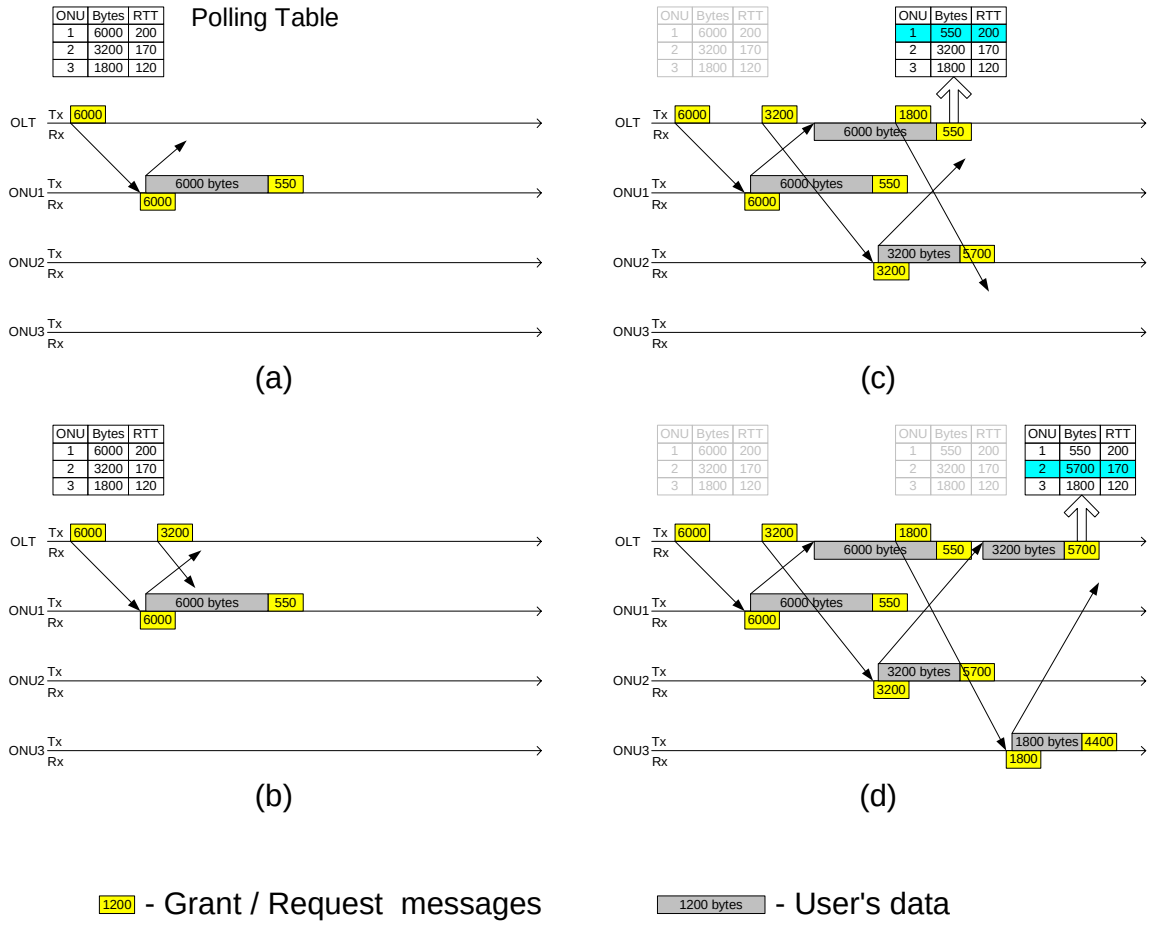


**Figure 5. Steps of the Polling Algorithm.**

If an ONU emptied its buffer completely, it will report 0 bytes back to the OLT. Correspondingly, in the next cycle the ONU will be granted 0 bytes, i.e., it will be allowed to send a new request, but no data.

Note that the OLT's receive channel is almost 100 % utilized (Requests and guard times consume a small amount of bandwidth). Idle ONUs (without data to send) are not given transmission windows. That leads to a shortened cycle time, which in turns results in more frequent polling of active ONUs.

As it is clear from the description above, there is no need to synchronize the ONUs to a common reference clock (as traditionally done in TDMA schemes). Every ONU executes the same procedure driven by the Grant messages received from the OLT. The entire scheduling and bandwidth allocation algorithm is located in the OLT. Thus, it is easy to adaptively change the

scheduling at run-time based on some network conditions; the ONUs do not need to negotiate or acknowledge new parameters, nor do they need to switch to new settings synchronously.

If the OLT authorizes each ONU to send its entire buffer contents in one transmission, ONUs with high data volume could monopolize the entire bandwidth. To avoid this, the OLT will limit the maximum transmission size. Thus, every ONU will get a Grant to send as many bytes as it has requested in a previous cycle, but no more than some maximum limit (maximum transmission window size). There could be various schemes for specifying the limit. It can be fixed, say, based on a Service Level Agreement (SLA) for each ONU, or dynamic - based on average network load. We will discuss different approaches of specifying the transmission window size in Section 3.5.

### 3.1 Control-Message Format

The Grant and Request messages should only contain two pieces of information: ONU's identification (NID: Node ID) and window size (WS). In Grant messages, the NID is used by an ONU to recognize a Grant destined to that ONU. The window size in a Grant is the size of the granted window. Requests contain a NID field to let the OLT know which table entry to update. The WS field in Requests is the number of bytes buffered in the ONU (think of it as a window size desired by the ONU).

In our initial design, we considered the control messages to be a data structure encapsulated in an Ethernet frame. The MAC address (either source or destination) in this case would serve as the ONU identification (NID). The payload would consist of only one field containing the window size. However, this approach has two problems:

**Problem 1: High downstream load and light upstream load**. If there is very light upstream traffic, each ONU will reply with its Request message and no or very little data. That means, that the OLT will poll the next ONU much sooner, i.e., Grants will be generated more often. This may result in lots of bandwidth being consumed by Grant messages in the downstream direction, where the load is heavy already.

**Problem 2: Blocking the Grant behind a long data packet**. If the OLT determines that the next Grant message should leave at time *t*, but the channel is busy transmitting a long data packet, the Grant message will be delayed, so will be delayed the transmission from the corresponding ONU, and thus more bandwidth will be lost.

If we were to keep the Ethernet frame format for the control messages, one solution to the above problems would be to move Grants to a separate downstream channel (say, to a separate wavelength). However, this would make the entire access network more expensive, as two transmitters will be needed in the OLT (one for downstream data channel and one for control channel) and two receivers will be required in every ONU.

Fortunately, another solution is available. First, we note that an Ethernet frame with its 64-byte minimum size is an overkill for a control message consisting only of 1-byte NID and 2-byte WS fields. The solution we propose is to embed the Grant messages inside the downstream data packets using escape sequences. To understand this approach, first recall that Gigabit Ethernet uses 8-to-10 bit encoding, i.e., every byte is being encoded as 10 bits before being sent over the medium (see IEEE standard 802.3). However, not all of 10-bit values are valid encoding of an 8-bit value. One or more of these "non-valid" codes can be chosen to represent an escape code (ESC). Thus, the control message (either Grant or Request) will look as shown in Figure 6.b.

Such a control message can be inserted in the middle of an Ethernet frame or between the frames (see Figure 6). The receiver will recognize the beginning of the control sequence by reading the ESC code. It will then extract the 3 bytes that follow the ESC byte before passing the rest of the received data to a standard Ethernet MAC.

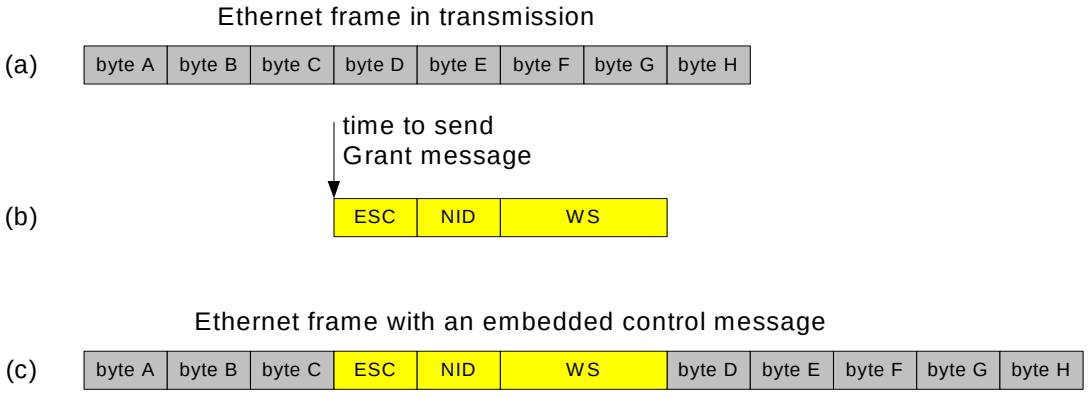


**Figure 6. Control message embedded in the middle of Ethernet frame:**
**(a) Ethernet frame; (b) control message (Grant);**
**(c) Ethernet frame with an embedded control message.**

Upstream control message (Requests) will use the same 4-byte format and will be sent at the beginning of the transmission from a given ONU (or, correspondingly, at the end of a guard

time). Due to the small size of control messages, we will not show them in the equations below; instead we will consider the Requests to be part of the guard time intervals.

### 3.2 Scheduling a Control Message

As is clear from the algorithm description, the Grant messages are being scheduled using the following formula:

$$G_j^{[i+1]} = MAX\begin{cases} G_j^{[i]} + r^{[i]} - r^{[i+1]} + \dfrac{W_j^{[i]}}{R_U} + B \\ G_{j-1}^{[i+1]} + r^{[i+1]} \end{cases} \qquad (11)$$

where

$G_j^{[i]}$ – time epoch when j$^{th}$ Grant to i$^{th}$ ONU is (will be) transmitted
( note that $G_j^{[i]} + r^{[i]}$ is a time epoch when j$^{th}$ Request from i$^{th}$ ONU is received )

$r^{[i]}$ – Round - Trip Time for the i$^{th}$ ONU

$W_j^{[i]}$ – j$^{th}$ Window Size for the i$^{th}$ ONU

$R_U$ – Transmission speed (bit rate)

$B$ – Guard Time (in $\mu$s)

Superscript in brackets identifies an ONU as following: $[\bullet] = (\bullet \text{ MOD } N) + 1$.

The top line in formula (11) says that we schedule the Grant to $i$+1$^{st}$ ONU such that its Request arrives after the guard time after the end of the transmission window from $i^{th}$ ONU (Figure 7). The bottom line in formula (11) says that the Grant cannot be sent before the previous Request from the same ONU is received, i.e., the interval between successive Grants to the same ONU is at least the round-trip time to that ONU. This is because the Grant needs information (requested window size) contained in the previous Request.

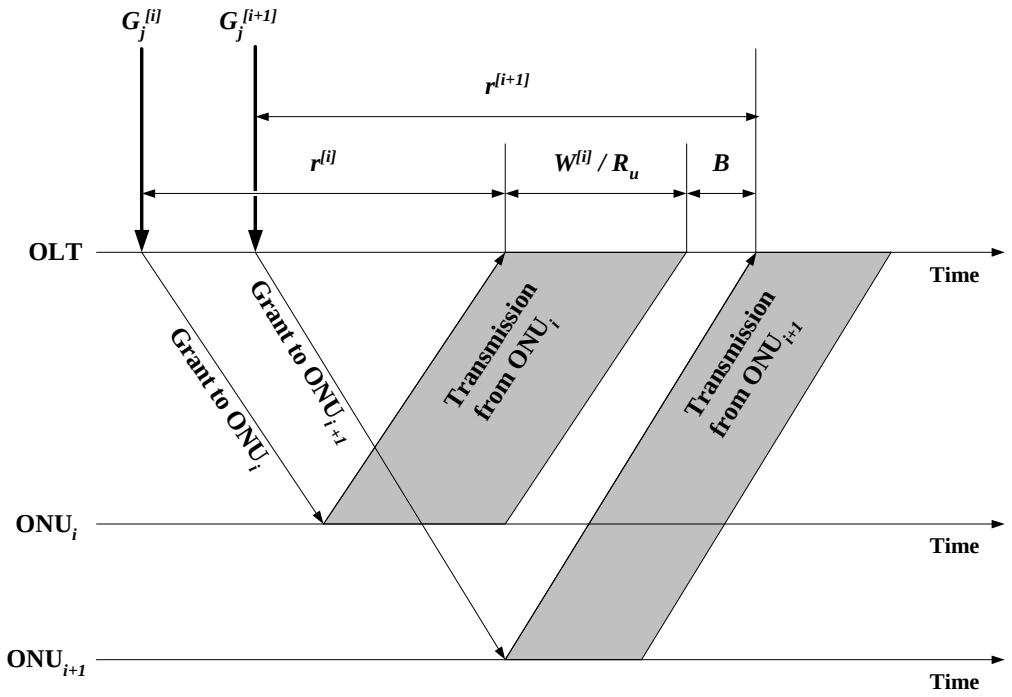


**Figure 7. Scheduling a Grant**

From formula (11) it is clear that under some circumstances the Grant to ONU$_{i+1}$ should be sent before the Grant to ONU$_i$. This will happen if

$$r^{[i+1]} > r^{[i]} + \frac{W_j^{[i]}}{R_U} + B \qquad (12)$$

In general, the order of Grants is different from the order of Requests. The Requests (and therefore, the data) from each ONU arrives in the same order (e.g., round robin) in every cycle. The Grants are scheduled with regard to the corresponding round-trip times and granted window sizes. As a result, the order of Grants may be different in every cycle. Scheduling the Grant to ONU *i*+1 ahead of the Grant to ONU *i* is not a problem as the order of Grants is determined in a cycle prior to when they should be sent out. There should be a clear distinction between operations of *scheduling* a Grant, i.e., determining a time in the future when the Grant is to be sent, and *sending* the Grant, i.e., transmitting it at the previously determined time.

Scheduling the Grants as described above may result in a Grant scheduling conflict. The conflict occurs when the absolute difference between the left and right parts of the inequality (12) is less than the transmission time of the Grant message. Adapting to the message format

described above and assuming transmission speed $R_U$ = 1 Gbps, the Grant transmission time is equal to 32 ns. To resolve such a conflict, the Grant that is scheduled last should be delayed till the end of transmission of the previous Grant. Grant delay has no significant effect on system performance; the only consequence of it is the corresponding delay of transmission from the ONU (i.e., increase of the guard time before that transmission). Of course, after the conflicting Grant is delayed, it may collide again with another Grant that was scheduled before. In the extreme case, some Grant may collide with at most $N$ - 1 other Grants, where $N$ is the number of ONUs. For the case of $N$ = 16, the maximum acquired delay is 0.48 μs. Obviously, this solution does not introduce any scalability issues in terms of the value of $N$ because multiple collisions only negligibly increase the guard time.

### 3.3 Disconnected ONU

During network operation, one or more ONU may become disconnected, either due to failure or loss of power. In FTTH systems, this may happen more often, as each customer has control over his/her ONU and may turn it off. A disconnected ONU should not use network resources. The proposed algorithm offers a simple solution to this problem: when the ONU gets disconnected, the next Grant message will timeout, i.e., the OLT will not receive a new Request within a timeout interval. The OLT will stop polling that ONU in every cycle. Instead, the OLT will poll that ONU much less frequently, say, once per minute. Of course, the OLT should distinguish between a corrupted (missing) Request and a disconnected ONU. If the OLT detected a transmission in a given window, but did not find a Request after the synchronizing preamble, it should still poll this ONU in the next cycle; however the granted window size should be zero. Only if no transmission was detected in the granted window should the OLT conclude that the ONU was disconnected.

When polling a disconnected ONU, we should not make any assumptions of the RTT value, i.e., we should not use the last known value. The reason for the ONU disconnection could be, for example, its relocation, and this means that the new RTT may be very different from the older one. The issue is how to schedule the Grant message to an ONU with an unknown RTT, or more generally, how to keep an interleaved schedule when some of RTTs are unknown. We may still use formula (11), although slightly modified, as follows:

a. If ONU $i$ is marked as disconnected, the Grant to the next ONU ($i$+1) will be scheduled as

$$G_j^{[i+1]} = MAX\begin{cases} G_j^{[i]} + TIMEOUT - r^{[i+1]} + B \\ G_{j-1}^{[i+1]} + r^{[i+1]} \end{cases} \quad (13)$$

b. If the OLT is to poll a disconnected ONU ($i$+1), it will schedule the Grant to that ONU as following:

$$G_j^{[i+1]} = MAX\begin{cases} G_j^{[i]} + r^{[i]} + B \\ G_{j-1}^{[i+1]} + r^{[i+1]} \end{cases} \quad (14)$$

Equation 13 assumes longest possible response time from a disconnected ONU. Equation 14 assumes shortest possible response time (zero response time) from a disconnected ONU, thus effectively preventing a pipelining with the previous ONU.

Figure 8 illustrates Grant scheduling for ONUs $i$, $i$+1, and $i$+2, where $ONU_{i+1}$ is marked as disconnected.

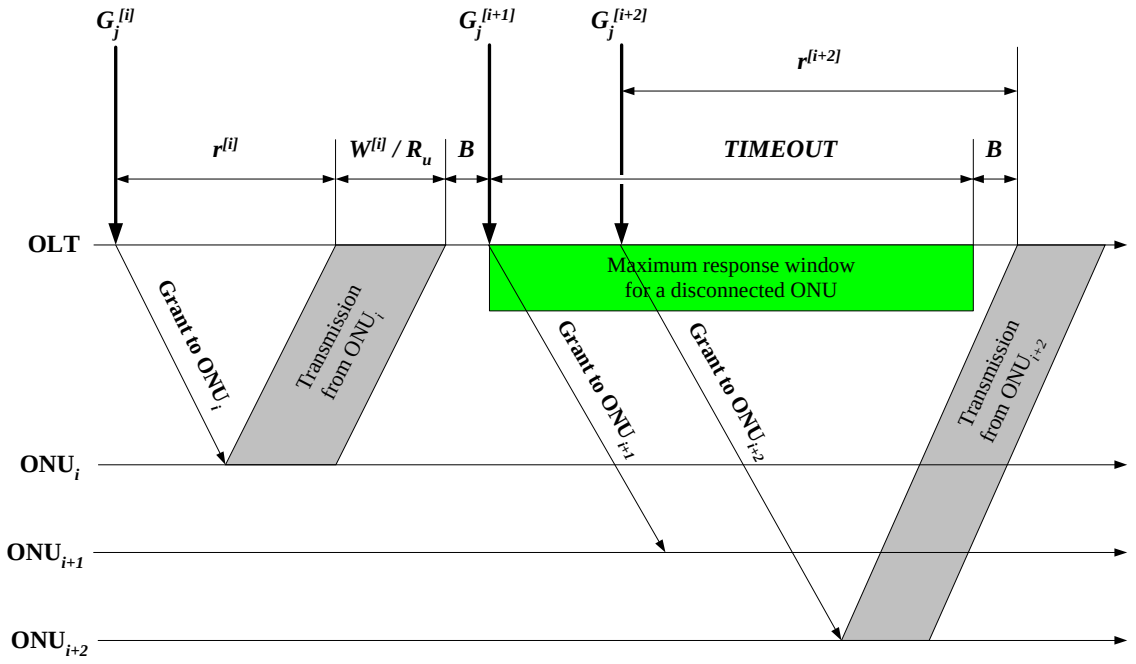


**Figure 8. Grant scheduling to ONU$_i$, ONU$_{i+1}$, and ONU$_{i+2}$, where ONU$_{i+1}$ is marked as disconnected.**

This approach guarantees two properties:

We will not send the Grant message to a disconnected ONU (with an unknown RTT) until we receive the entire transmission from the previous ONU. Thus, a disconnected ONU that suddenly came alive with a very small RTT will not collide with the previous ONU.

The OLT will be able to schedule the Grant to the next ONU (i.e., keep its pipeline full) even if the current ONU is still disconnected. The reason is that the reply can only be received within the TIMEOUT interval. Thus, the data from the next ONU can safely arrive after time TIMEOUT after sending the Grant to a disconnected ONU, no matter whether the disconnected ONU has come alive or not.

If we do receive a Request from an ONU previously marked as disconnected, we mark it as alive and begin polling it in every cycle. If we do not receive a Request within a TIMEOUT interval, the ONU remains disconnected, i.e., next time it will be polled after a 1-minute interval.

### 3.4 Cold Start

In the above algorithm, we started with the OLT already having its table populated. Here we describe how to fill the table at cold start. At cold start, the OLT assumes all ONUs to be disconnected. As such, for every ONU, the OLT performs the polling procedure for disconnected ONU described above. Given $N$ ONUs, the total time to perform the cold start procedure is $N \times$ TIMEOUT.

### 3.5 Maximum Transmission Window

To prevent the upstream channel monopolization by one ONU with high data volume, there should be a maximum transmission window size limit assigned to every ONU. We denote an ONU-specific maximum transmission window size $W_{MAX}^{[i]}$. The choice of specific values of $W_{MAX}^{[i]}$ determines the maximum polling cycle time $T_{MAX}$ under heavy load conditions:

$$T_{MAX} = \sum_{i=1}^{N} \left( B + \frac{W_{MAX}^{[i]}}{R_U} \right) \tag{15}$$

Making $T_{MAX}$ too large will result in increased delay for all the packets, including high-priority (real-time) packets. Making $T_{MAX}$ too small will result in more bandwidth being wasted by guard times.

It is ONU's responsibility to make sure that the packet it is about to send fits in the remainder of the granted window. If the packet does not fit, it should be deferred till the next Grant arrives. In this case the transmission window may remain underutilized (refer to [4] for a discussion on transmission window utilization).

In addition to the maximum cycle time, the $W_{MAX}^{[i]}$ value also determines the guaranteed bandwidth available to ONU-*i*. Let $\Lambda_{MIN}^{[i]}$ denote the guaranteed bandwidth of ONU i. Obviously,

$$\Lambda_{MIN}^{[i]} = \frac{W_{MAX}^{[i]}}{T_{MAX}} \tag{16}$$

i.e., the ONU is guaranteed to be able to send $W_{MAX}^{[i]}$ bytes in at most $T_{MAX}$ time. Of course, an ONUs bandwidth will be limited to its guaranteed bandwidth only if all other ONUs in the system also use all their available bandwidth. If at least one ONU has less data, it will be granted a shorter transmission window, thus making the cycle time shorter, and therefore the available bandwidth to all other ONUs will increase proportionally to their $W_{MAX}^{[i]}$. This is the mechanism behind dynamic bandwidth distribution: by adapting the cycle time to the instantaneous network load (i.e., queue occupancy) the bandwidth is automatically distributed to ONUs based on their loads. In the extreme case, when only one ONU has data to send, the bandwidth available to that ONU will be:

$$\Lambda_{MAX}^{[i]} = \frac{W_{MAX}^{[i]}}{N \times B + \frac{W_{MAX}^{[i]}}{R_U}} \tag{17}$$

In our simulations, we assumed all ONUs having same SLA, i.e., $W_{MAX}^{[i]} = W_{MAX}$, $\forall i$. That resulted in

$$T_{MAX} = N\left(B + \frac{W_{MAX}}{R_U}\right) \tag{18}$$

We believe $T_{MAX}$ = 2 ms and $B$ = 5 μs are reasonable choices. That made $W_{MAX}$ = 15000 bytes. With that choice of parameters, every ONU will get a guaranteed bandwidth of 60 Mbps, and maximum (best-effort) bandwidth of 600 Mbps (see equations 16 and 17).

Before discussing the simulation results, let us take a look at the components of the packet delay (Figure 9).

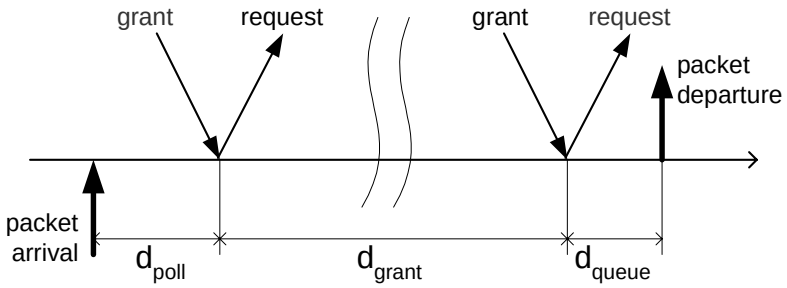


**Figure 9. Components of packet delay.**

The packet delay $d$ is equal to:

$$d = d_{poll} + d_{grant} + d_{queue} \tag{19}$$

where

$d_{poll}$ - time between packet arrival and next Request sent by that ONU. On average

$$d_{poll} = T / 2 \tag{20}$$

$d_{grant}$ - time interval from ONU's request for a transmission window for the packet till the Grant from OLT received. This delay may span multiple polling cycles, depending on how many packets were in the queue at the time of the new arrival. In general, denoting $q$ to be the queue size (including the new packet size) at the moment of new packet arrival, and $W_P^{[i]}$ - the pending Grant size (i.e., requested before the new packet arrived), we have

$$d_{grant} = T \times \left\lceil \frac{q - W_P^{[i]}}{W_{MAX}} \right\rceil \tag{21}$$

$d_{queue}$ - queuing delay after the corresponding Grant from the OLT arrived. This delay is insignificant comparing to the previous two.

$$d_{queue} = \begin{cases} \dfrac{q}{R_U} & , \quad q \le W_P^{[i]} \\ \dfrac{(q - W_P^{[i]}) \bmod W_{MAX}}{R_U} & , \quad q > W_P^{[i]} \end{cases} \tag{22}$$

The remaining question is how the OLT should determine the granted window size if the requested window size $W^{[i]} < W_{MAX}$ ? Table 1 defines a few approaches (services) the OLT may take in making its decision:

| Service | Formula | Description |
|---|---|---|
| Fixed | $W^{[i]} = W_{MAX}$ | This scheduling discipline ignores the requested window size and always grants the maximum window. As a result it has a constant cycle time $T_{MAX}$. Essentially this approach corresponds to the PON system described in [4]. It is shown here only for comparison. |
| Limited | $W^{[i]} = MIN\begin{cases} V^{[i]} \\ W_{MAX} \end{cases}$<br>$V^{[i]}$ - requested window size | This discipline grants the requested number of bytes, but no more than $W_{MAX}$. It is the most conservative scheme and has the shortest cycle of all the schemes. |
| Gated | $W^{[i]} = V^{[i]}$ | This service discipline does not impose the $W_{MAX}$ limit on the granted window size, i.e., it will always authorize an ONU to send as much data as it has requested. Of course, without any limiting parameter, the cycle time may increase unboundedly if the offered load exceeds the network throughput. In this discipline, such a limiting factor is the buffer size $Q$, i.e., an ONU cannot store more than $Q$ bytes, and thus, it will never request more than $Q$ bytes. |
| Constant Credit | $W^{[i]} = MIN\begin{cases} V^{[i]} + Const \\ W_{MAX} \end{cases}$ | This scheme adds a constant credit to the requested window size. The idea behind adding the credit is the following: assume $x$ bytes arrived between the time when an ONU sent a Request and received the Grant. If the granted window size equals requested window + $x$ (i.e., it has a credit of size $x$), then the $d_{grant}$ delay component will be zero for these $x$ bytes and the total delay will be shorter. |

| | | |
|---|---|---|
| Linear Credit | $W^{[i]} = MIN\begin{cases} V^{[i]} \times Const \\ W_{MAX} \end{cases}$ | This scheme uses a similar approach as the Constant Credit scheme. However, the size of the credit is proportional to the requested window. The reasoning here is the following: LRD traffic possesses a certain degree of predictability (see [9]), viz., if we observe a long burst of data, then this burst is likely to continue for some time into the future. Correspondingly, the arrival of more data during the last cycle time may signal that we are observing a burst of packets. |
| Elastic | $W^{[i]} = MIN\begin{cases} V^{[i]} \\ NW_{MAX} - \sum_{j=i-N}^{i-1} W^{[j]} \end{cases}$ | Elastic service is an attempt to get rid of a fixed maximum window limit. The only limiting factor is the maximum cycle time $T_{MAX}$. The maximum window is granted in such a way that the accumulated size of last $N$ grants (including the one being granted) does not exceed $NW_{MAX}$ bytes ($N$ – number of ONUs). Thus, if only one ONU has data to send, it may get a Grant of size up to $NW_{MAX}$. |

**Table 1. Grant scheduling services used in simulation.**

## 4 Simulation Results

In Figure 10, we present the mean packet delay for different Grant scheduling services as a function of an ONU's offered load $\phi$. In this simulation, all ONUs had identical load; therefore, the offered network load $\Phi$ is equal $N\phi$.

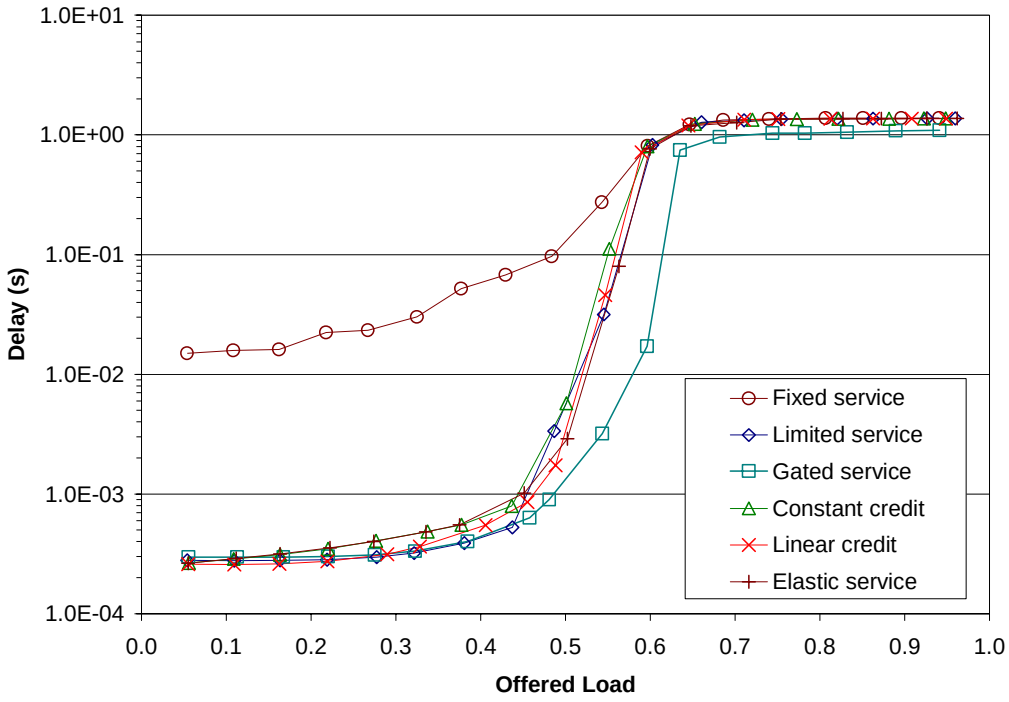


**Figure 10. Mean packet delay.**

As can be seen in the figure, all granting services except fixed and gated have almost coinciding plots. We will discuss fixed and gated service results below. As for the rest of them, no other method gives a detectable improvement in packet delay. The explanation to this lies in the fact that all these methods are trying to send more data by way of increasing the granted window size. While that may result in a decrease or elimination of the $d_{grant}$ delay component for some packets, it will increase the cycle time, and thus result in an increase of the $d_{poll}$ component for all the packets.

The fixed service plot is interesting as an illustration of the traffic long-range dependence. Even at the very light load of 5%, the average packet delay is already very high (~15ms). This is because most packets arrive in very large packet trains. In fact, the packet trains were so large that the 10-Mbyte buffers overflowed and about 0.14% of packets were dropped. Why do we observe this anomalous behavior only with fixed service? The reason is that the other services have a much shorter cycle time; there is just not enough time in a cycle to receive more bytes than $W_{MAX}$, thus the queue never builds up. In fixed service, on the other hand, the cycle is large (fixed) from the very beginning and several bursts that arrived close to

each other can easily overflow the buffer. It can be shown, that, in order to have buffer overflow, a time interval of length $\Delta t$ should have an average arrival rate $\lambda$, such that

$$\lambda \geq \frac{Q}{\Delta t} + \frac{W_{MAX}}{T_{MAX}} \tag{23}$$

Given our numerical values for $Q$, $W_{MAX}$, and $T_{MAX}$, buffer overflow will happen if, for example, we have a 2.5-second interval with average arrival rate of 92 Mbps, or a 4-second interval with the rate of 80 Mbps, etc. And because the traffic is heavy-tailed, we do encounter such intervals.

We want to note here that the reduced cycle time which adapts exactly to the amount of data available in the ONUs is the main advantage of the proposed algorithm. Before we continue with our discussion of gated service, we would like to present the simulation result for the average queue size (Figure 11).

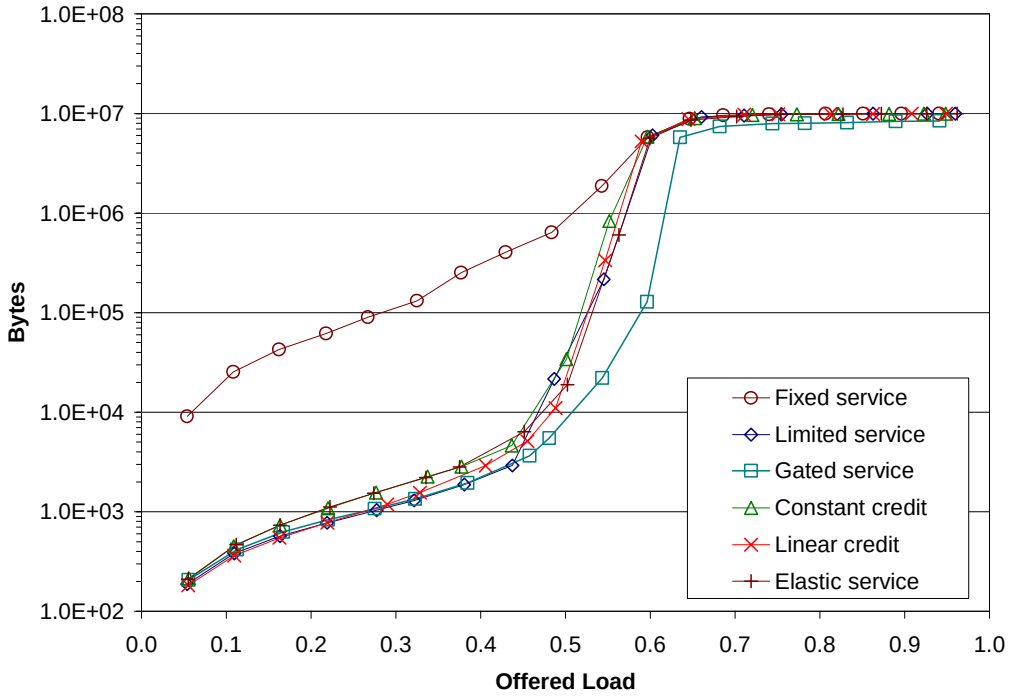


**Figure 11. Average queue size.**

This picture is similar to the mean delay plot. Again, fixed service has a larger queue, which comes as no surprise.

Let us now turn our attention to the delay and queue size plots for gated service. It can be noticed that gated service provides a considerable improvement in the mid-range load between 45% and 65%. At 60% load, for example, the delay and average queue size are approximately 40 times less than with other services. This happens because the gated service has higher channel utilization due to the fact that the cycle time is much larger, and, as a result, fewer guard times are used per unit of time. For the same reason, its saturation delay is a little bit lower than in other services (refer to Figure 10) – the entire buffer contents are being transferred in one window rather then in batches of $W_{MAX}$ bytes with a guard time in front of each batch.

Next, we will show that, even though gated service has lower delay and average queue size, it is not a suitable service for an access network under consideration. The problem lies in the much longer cycle (see Figure 12). As a result, the $d_{poll}$ delay will be much larger and therefore, the packet latency will be much higher. Clearly, large $d_{grant}$ and $d_{queue}$ delay components can be avoided for high-priority packets by using priority queuing. But $d_{poll}$ is a fundamental delay, which cannot be avoided in general. This makes gated service not feasible for interleaved polling access network.

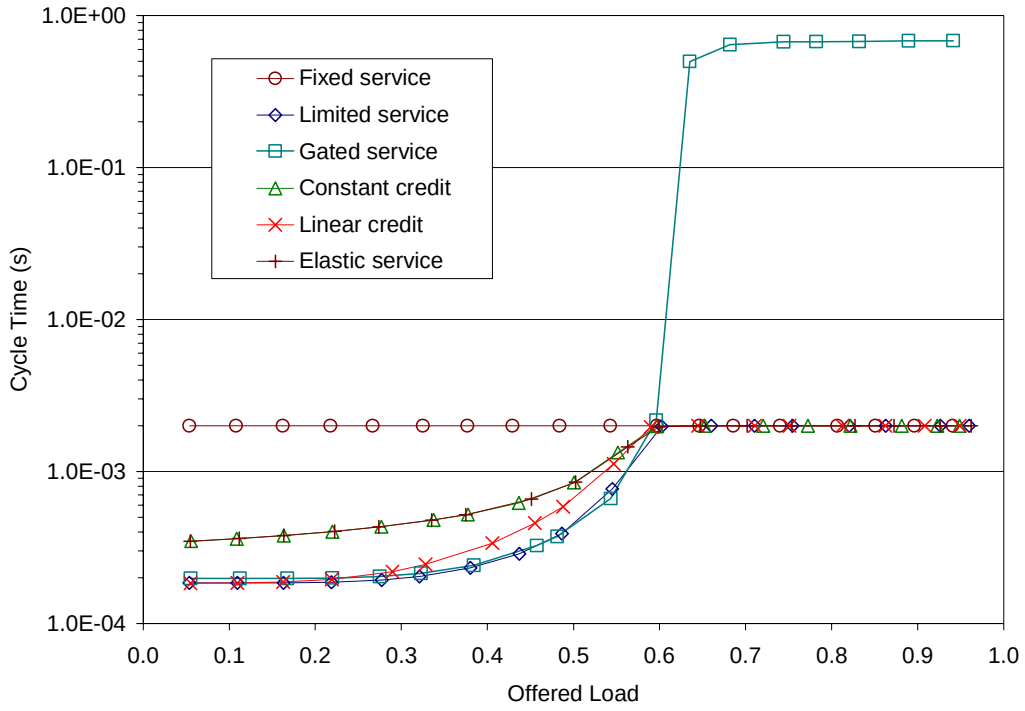


**Figure 12. Mean cycle times for various service disciplines**

Thus, we conclude that neither of the discussed service disciplines is better than limited service. As such, for the remainder of this study we will focus our attention on the limited service discipline. In the next section, we will analyze the fairness and quality of service characteristics of limited service.

### 4.1 Performance of Limited Service

In this section, we analyze the performance of a tagged ONU *i* as a function of its offered load ($\phi_i$) and the effective load of the entire network ($\Omega$). In Figure 13, we present the average packet delay.

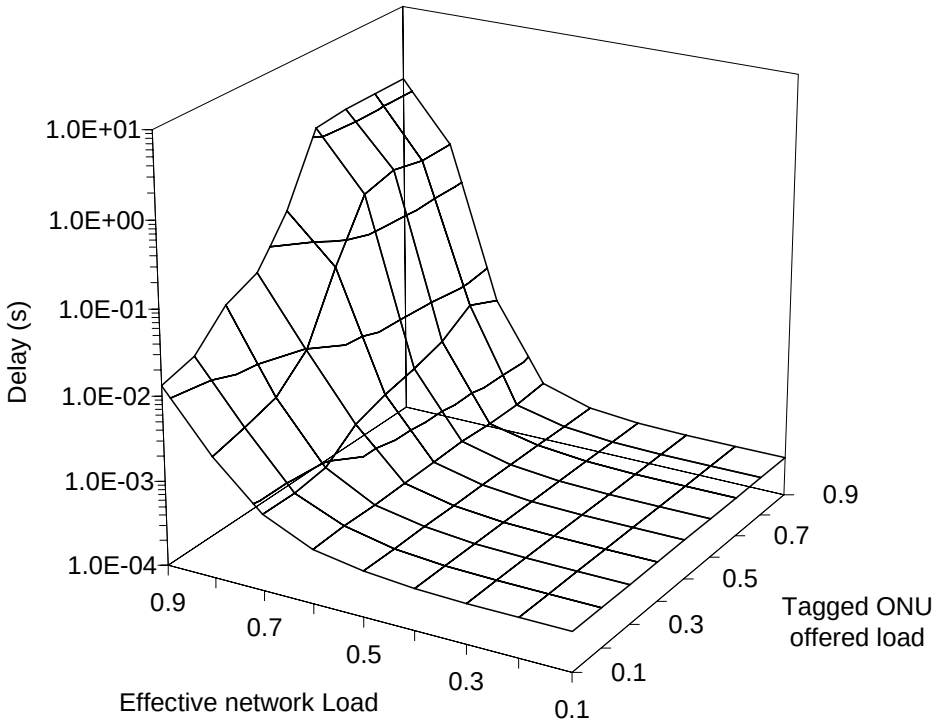


**Figure 13. Average packet delay as a function of effective network load and ONU offered load.**

When the effective network load is low, all packets in a tagged source experience very little delay, no matter what the ONU's offered load is. This is a manifestation of dynamic bandwidth allocation – when the network load is low, the tagged source gets more bandwidth.

The opposite situation – low offered load at the ONU and high effective network load - results in a higher delay. The only reason to this is the burstiness (i.e., long-range dependence) of the traffic. This is the same phenomenon observed with fixed service: high network load results in increased cycle time. That cycle time is large enough to receive more than $W_{MAX}$ bytes of data during a burst. Hence, the $d_{grant}$ delay for some packets will increase beyond one cycle time. We will discuss a way to combat this phenomenon by using priority queuing in Section 5.2.

Figure 14 shows the probability of a packet loss in a tagged ONU *i* as a function of its offered load ($\phi_i$) and the effective load of the entire network ($\Omega$).

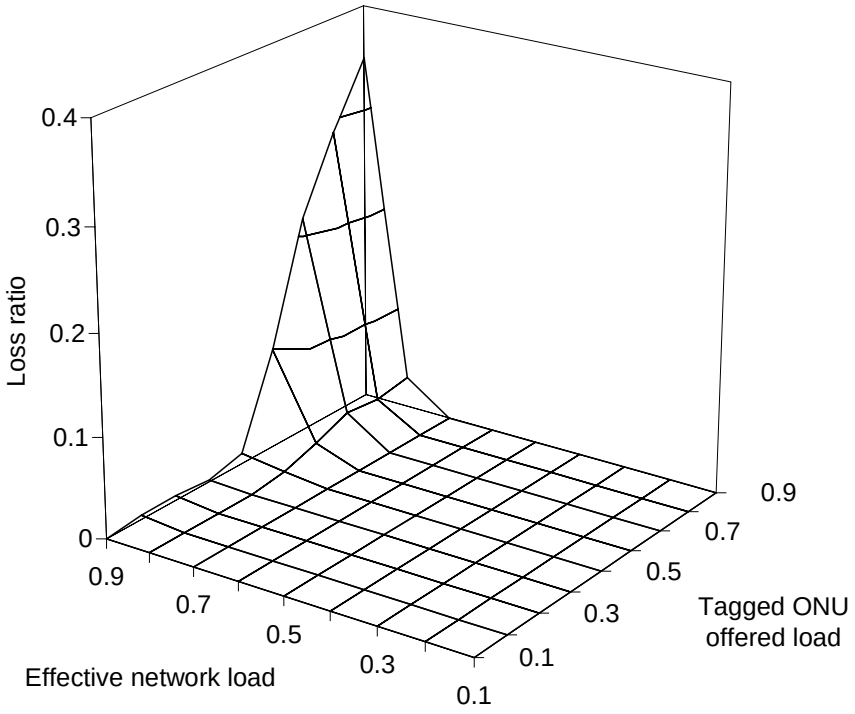


**Figure 14. Packet loss ratio as a function of effective network load and ONU offered load.**

Once again we observe that packet loss is zero or negligible if the effective network load is less than 80%. When the network load is above 80% and the tagged ONU offered load is above 50%, we observe considerable packet loss due to buffer overflow.

## 4.2 QoS Considerations

As a full-service access network, the proposed architecture should support a multitude of services, i.e., best-effort data, variable-bit-rate (VBR) video stream, constant-bit-rate (CBR) stream (for legacy equipment like Plain Old Telephone Service (POTS) lines, Private Branch Exchange (PBX) boxes), etc.

In this section, we will investigate how priority queuing will allow us to provide a delay bound for some services. We classify our data in three priority classes:

*Best Effort* (BE) class has the lowest priority. This priority level is used for non-real time data transfer. There is no delivery or delay guarantees in this service. The BE queue in the ONU is served only if higher priority queues are empty. Since all queues in our system share the same buffer, the packets arriving to higher priority queues may displace the BE packets that are already in the BE queue. In our experiment, the tagged source has the BE traffic with an average load of 0.4 (40 Mbps).

*Assured Forwarding* (AF) class has higher priority than the BE class. The AF queue is served before the BE queue. In our experiment, the AF traffic consisted of a VBR stream with average bit rate of 16 Mbps. This corresponds to three simultaneous MPEG-2-coded video streams [13]. Since the AF traffic is also highly bursty (LRD), it is possible that some packets in long bursts will be lost. This will happen if the entire buffer is occupied by AF or higher priority packets.

*Guaranteed Forwarding* (GF) priority class was used to emulate a T1 line in the packet-based access network. The GF class has the highest priority and can displace the BE and AF data from their queues if there is not enough buffer space to store the GF packet. A new GF packet will be lost only if the entire buffer is occupied by GF packets. The GF queue is served before the AF and BE queues. The T1 data arriving from the user is packetized in the ONU by placing 24 bytes of data in a packet. Including Ethernet and UDP/IP headers results in a 70-byte frame generated every 125 μs. Hence, the T1 data consumed the bandwidth equal to 4.48 Mbps. Of course, we could put 48 bytes of T1 data in one packet and send one 94-byte packet every 250 μs. This would consume only 3.008 Mbps, but will increase the packetization delay.

Figures 15 and 16 show the average and the maximum packet delay for each type of traffic. The average load of the tagged ONU was set to 40 Mbps of BE data, 16 Mbps of AF data, and 4.48 Mbps of GF data, or to a total of ~60 Mbps. The horizontal axis shows the effective

network load. This figure investigated how the traffic parameters depend on the overall network load.

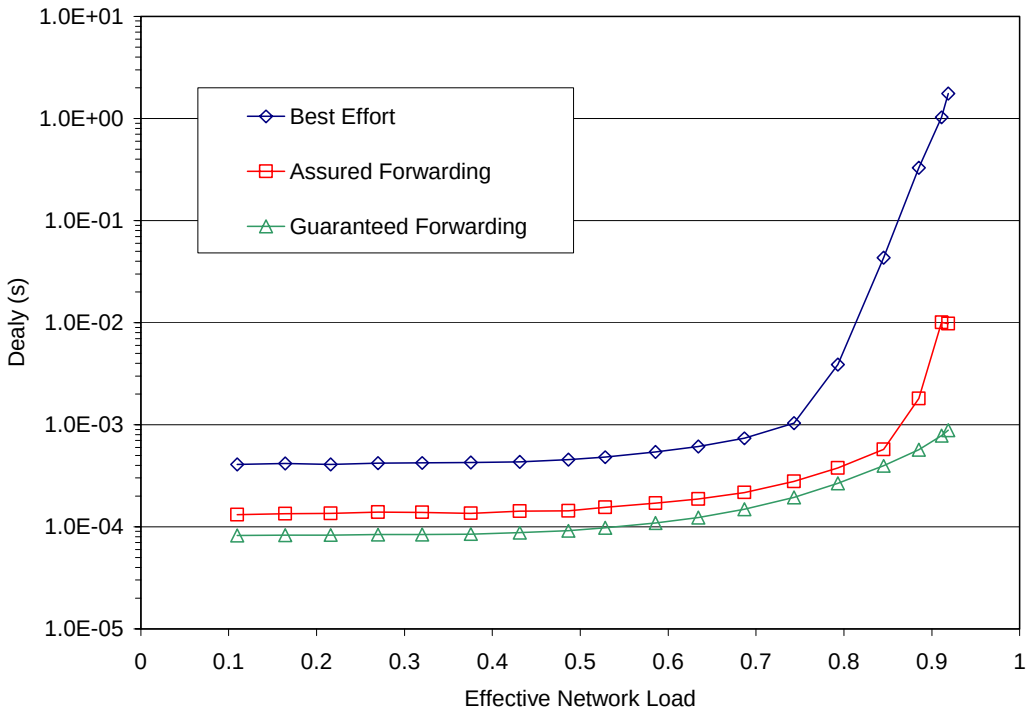


**Figure 15. Average packet delay for various classes of traffic as a function of effective network load.**

We can see that the BE traffic suffered the most when the ambient load increased. Its delay increased and the simulation results showed that some packets were discarded when network load exceeded 80%. The AF data also experienced an increased delay, but no packet losses were observed. The increased delay in the AF traffic can be attributed to long bursts of data. Clearly, applying some kind of traffic shaping/policing limiting the burst size at the source would improve the situation. The GF data experiences a very slight increase in both average and maximum delays. This is due to the fact that the packets were generated with a constant rate, i.e., no data bursts. The average delay in this case exactly followed the average cycle time, being one half of that. The maximum delay is equal to the maximum observed cycle time and for any effective network load is bounded by $T_{MAX}$ (2 ms with our chosen set of parameters).

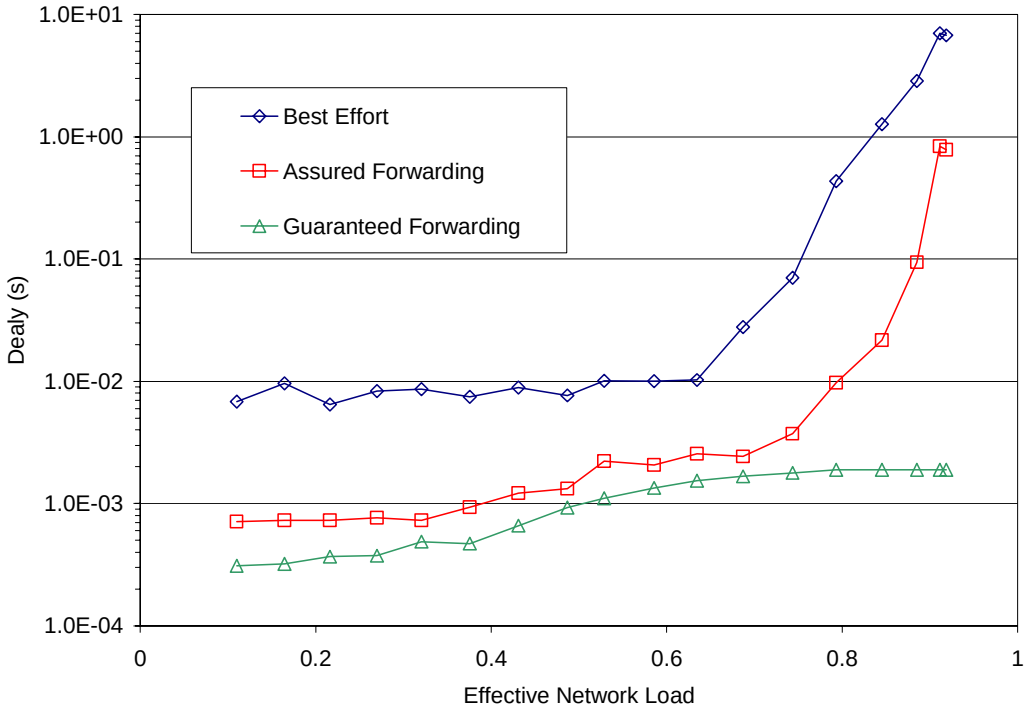


**Figure 16. Maximum packet delay for various classes of traffic as a function of effective network load.**

Of course, to restore the proper T1 rate, a shaping buffer (queue with constant departure rate) should be employed at the receiving end (in OLT). After receiving a packet (or a group of packets) from an ONU, the shaping buffer should have at least 2 ms worth of data, i.e., 384 bytes of T1 data. This is because the next packet from the same ONU may arrive after the maximum delay of 2 ms. When such a delayed packet arrived, it still should find the non-empty buffer. Let us say the minimum buffer occupancy is 24 bytes which is 125 μs of T1 transmission time (we call it a *buffer under-run protection time*). Thus, in this case, the overall latency experienced by T1 data will consist of

1. 125 μs of packetization delay in ONU;
2. polling delay in ONU (up to $T_{MAX}$);
3. up to 100 μs of propagation delay (assuming maximum distance of 20 km);
4. Wait time in the shaping buffer.

Items 2 and 4 together are equal to $T_{MAX}$ plus buffer under-run protection time, i.e., 2.125 ms. Thus, the total latency is about 2.35 ms. If this latency is too high for a given specification, it can be reduced by decreasing $T_{MAX}$ (see equation 19).

# 5 Conclusion

In this study, we discussed and evaluated design issues that must be dealt with in a PON access network. Specifically, to drive the cost of an access network down, it is very important to have an efficient, scalable solution. We believe that a PON based on polling and with data encapsulated in Ethernet frames possesses the best qualities, such as use of a single downstream and a single upstream wavelength, ability to provision a fractional wavelength capacity to each user, and ease of adding a new user. We showed the advantages of using Ethernet instead of ATM to carry IP data.

We proposed a simple algorithm for dynamic bandwidth allocation based on an interleaved polling scheme with an adaptive cycle time. We suggested a novel approach for an in-band signaling that allowed using a single wavelength for both downstream data and Grants transmission. Also, we showed this approach to be scalable with the number of ONUs in the system.

Since each ONU uses the window size that is required at the moment, the polling cycle time *adapts* to the instantaneous queue loads, leading to an adaptive cycle time. This is the basic idea behind the fair unused bandwidth redistribution: reduced cycle time leads to an increase in the amount of best-effort bandwidth available to busy ONUs. This increase is proportional to their guaranteed bandwidth values. We found that, by limiting the maximum cycle time (when all ONUs are using their guaranteed bandwidths), we can provide hard limits on the delay and jitter value for CBR traffic. This is critical for a PON's ability to support legacy circuit-based equipment.

We also showed that the guaranteed bandwidth available to a user could easily be re-provisioned by simply changing a single parameter ($W_{MAX}$). In our simulation, we found that a disconnected ONU consumes a negligible amount of channel bandwidth, only about 0.0005%.

An interesting future research would be to investigate how the proposed algorithm responds to a network-wide synchronization (phasing effects) – a phenomenon where burst of data from independent flows tend to synchronize in time. This may have a detrimental effect on gains achieved by using any statistical multiplexing scheme, and IPACT in particular. At this time very little research is done in this area.

## Bibliography


[1] G. Pesavento and M. Kelsey, "*PONs for the broadband local loop*," Lightwave, PennWell, vol. 16, no. 10, pp. 68 – 74, September 1999.

[2] B. Lung, "*PON architecture 'futureproofs' FTTH*," Lightwave, PennWell, vol. 16, no. 10, pp. 104 – 107, September 1999.

[3] S. Hardy, "*Verizon staffers find fiber-to-the-home cheaper than copper*," Lightwave, PennWell, vol. 17, no. 134 pp. 1, December 2000.

[4] G.Kramer, B. Mukherjee, and G.Pesavento, "*Ethernet PON (ePON): Design and Analysis of an Optical Access Network*," Photonic Network Communications, vol. 3, no. 3, pp. 307-319, July 2001.

[5] W. Leland, M. Taqqu, W. Willinger, and D. Wilson, "*On the Self-Similar Nature of Ethernet Traffic (Extended Version*)," IEEE/ACM Transactions on Networking, 2(1), pp. 1-15, February 1994.

[6] S. Clavenna, "*Metro Optical Ethernet*," Lightreading (www.lightreading.com), November 2000.

[7] K. Claffy, G. Miller, and K. Thompson, "*The nature of the beast: Recent traffic measurements from an internet backbone*," in Proceedings INET '98, (Geneva, Switzerland), July 1998. Available at http://www.isoc.org/inet98/proceedings/6g/6g_3.htm.

[8] W. Willinger, M. S. Taqqu, and A. Erramilli, *A bibliographical guide to self-similar traffic and performance modeling for modern high-speed networks*, in Stochastic Networks, F. P. Kelly, S. Zachary, and I. Ziedins (eds.), Oxford University Press, Oxford, pp. 339-366, 1996.

[9] K. Park and W. Willinger, *Self-similar network traffic: An overview*, In K. Park and W. Willinger, editors, Self-Similar Network Traffic and Performance Evaluation. Wiley Interscience, 2000.

[10] A. Adas, "*Traffic models in broadband networks*," IEEE Communications Magazine, 35(7), pp. 82--89, July 1997.

[11] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson. "*Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level*," In Proc. ACM SIGCOMM '95, pp. 100-113, Cambridge, MA, August 1995.

[12] J. L. Hammond and P. J. P. O'Reilly. *Performance Analysis of Local Computer Networks*, Addison Wesley, 1987.

[13] M. W. Garrett and W. Willinger, "*Analysis, Modeling and Generation of Self-Similar VBR Video Traffic*," Proc. ACM Sigcomm'94, London, pp. 269-280, September 1994.