Design, Implementation, and Evaluation of an XG-PON Module for ns-3 Network Simulator

Simulation: Transactions of the Society for Modeling and Simulation International
XX(X):1–15

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Jerome A. Arokkiam[1,2], Pedro Alvarez[2], Xiuchao Wu[1], Kenneth N. Brown[1], Cormac J. Sreenan[1], Marco Ruffini[2], Nicola Marchetti[2], Linda Doyle[2], David Payne[2]

Abstract

10-Gigabit-capable Passive Optical Network (XG-PON), one of the latest standards of optical access networks, is regarded as one of the key technologies for future Internet access networks. This paper presents the design and evaluation of our XG-PON module for the ns-3 network simulator. This module is designed and implemented with the aim to provide a standards-compliant, configurable, and extensible module that can simulate XG-PON with reasonable speed and support a wide range of research topics. These include analysing and improving the performance of XG-PON, studying the interactions between XG-PON and the upper-layer protocols, and investigating its integration with various wireless networks. In this paper, we discuss its design principles, describe the implementation details, and present an extensive evaluation on both functionality and performance.

Keywords

XG-PON, ns-3, simulation, modelling, performance, evaluation

Introduction

Passive Optical Networks (PON) offer a highly-efficient and cost-effective technology for broadband access, based on a model of point-to-multipoint distribution using passive components.

PONs gained their popularity with the standardisation of gigabit-capable Ethernet PON (EPON) by Ethernet in the First Mile (EFM) task force[14] of the Institute of Electrical and Electronics Engineers (IEEE) in 2004; Full Service Access Network (FSAN) group of the International Telecommunications Union (ITU) also introduced its version of gigabit-capable PON standard with Gigabit-capable PON (GPON)[17] shortly after, in 2005. Since then, FTTx (Fibre To The Home/Building/Curb, etc.) networks based on standardised PON technologies, have been widely deployed in many countries around the globe, including in the US, Korea and Japan.

10-Gigabit-capable Passive Optical Network (XG-PON) is the new standard released by the FSAN. XG-PON improves GPON in numerous aspects; notable changes include increasing the default downstream data rate to 10 Gb/s, while increasing the upstream data rate to 2.5 or 10 Gb/s; the minimum logical split is increased to 256 (from 64 in GPON) and the physical reach extended up to 60 Km.

Since XG-PON could pave the way for many bandwidth-intensive applications (IPTV, Video On Demand, Video Conference, etc.), it is very important to study the performance issues arising with the deployment of XG-PON. For instance, it is valuable to study the impacts on the performance of XG-PON, when the propagation delay is much longer than that of the current PON networks[30]. It is also important to investigate the interactions between XG-PON and the upper-layer protocols (TCP[28], etc.) for improving user experience[16]. In addition, XG-PON has been proposed for Fibre To The Cell, in which XG-PON acts as the backhaul for multiple base stations of a cellular network[18]. Under this scenario, it is also very valuable to study its integration with various wireless networks (LTE[3], WiMAX[15], etc.) for providing high speed mobile Internet access.

Due to its very recent standard definition, XG-PON is still in its early stages of deployment - thus the above research topics can only be studied through simulation as it is too expensive and highly complex to set-up a full-fledged and fine-grained XG-PON test-bed. In this paper, we present an XG-PON module for the state-of-the-art ns-3[1] network simulator. Our XG-PON module is based on a series of G.987 Recommendations from the FSAN group of ITU. These recommendations mainly define the specifications of Physical Media Dependent (PMD) and Transmission Convergence (TC) layers of XG-PON. To study the above research topics with reasonable simulation speed, the optical distribution network and the operations of the physical layer are simplified significantly. This XG-PON module focuses on the issues of TC layer, such as frame structure, resource allocation, Quality of Service (QoS) management, and Dynamic Bandwidth Assignment (DBA) algorithms for the upstream wavelength. During the

[1]Dept. of Computer Science, University College Cork (UCC), Ireland
[2]Trinity College Dublin (TCD), Ireland

Corresponding author:
Jerome A. Arokkiam, Room 2.05, CONNECT Centre for Future Networks, Dunlop Oriel House, Trinity College Dublin, Dublin 2, Ireland
Email: a.jerome@tcd.ie

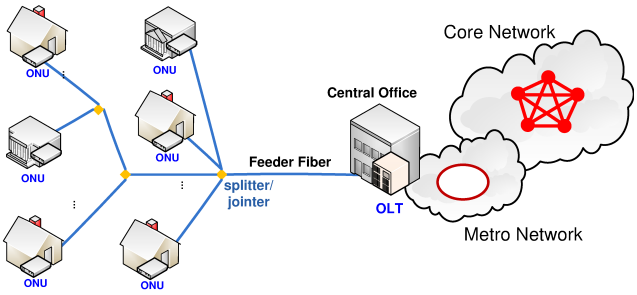

Figure 1. An illustration of PON as part of regional network

design and implementation of this module, we have also paid a lot of attention to its extensibility and reconfigurability. Since this XG-PON module needs to simulate a 10Gb/s network and hundreds of Optical Network Units (ONU), its performance (the speed of simulation and the overhead of memory) has also been given a high priority when designing and implementing these components.

This is the first standards-compliant XG-PON module designed for an open-source simulation platform. It allows reasonable simulation speed, even when the XG-PON supports more than 1000 ONUs and is simulated at 9.6Gb/s in the downstream (validated by our experiments in 'Evaluation results' section). This simulation platform enables extensive study of the performance issues that would arise with practical deployment of XG-PON, and allows applications of and modifications to XG-PON, to be explored in a common and open-source simulation environment. Our XG-PON module is built completely in C++ with 72 classes and approximately 22,000 lines of code. This code is under the GNU General Public License and can be downloaded through sourceforge[2]. More related information can be found on our websites (CTVR[1] and MISL[2]).

Organisation of the paper is such that, firstly PON, ns-3, and related works are briefly introduced. Then the details of XG-PON are presented. The design principles and key decisions are discussed thereafter, followed by the details on the trade-offs made in terms of simulation accuracy and speed. After that more specific details of the design and implementation of our XG-PON module for ns-3 are given. Evaluation results on both functionality and performance are then demonstrated before concluding the paper along with directions for future work.

Background and related work

Passive Optical Network (PON)

Compared to copper, optical fibre can provide higher bandwidth over a longer distance; but deployment of the latter in access networks is severely hindered by its heavy capital (capex) and operational (opex) expenditure. Hence PON was specifically designed to reduce overall cost by sharing fibre and electronic equipment among multiple users. This concept of shared optical medium in PON heavily improved the economic viability of FTTx.

As shown in Figure 1, PON is a point-to-multipoint fibre network and there are two kinds of equipment: the first kind, the active elements, includes the OLT (Optical Line Terminal) in central office and ONUs at/near customer premises; the second kind, a passive element, includes the optical splitters/jointers, which connect OLT and ONUs, without the need for any other active electronic equipment in the middle, thereby reducing capex and opex significantly.

In a classical TDMA (Time Division Multiple Access) based PON network, downstream traffic is broadcast by the OLT to all ONUs that share the same optical fibre and encryption is used to prevent eavesdropping. Upstream traffic from ONUs is interleaved by OLT to use the optical fibre in a TDMA-like manner. Since ONUs normally have different distances to the OLT, the data bursts from these ONUs must be scheduled carefully to provide a collision-free and efficient upstream data communication. To accommodate the dynamics in bandwidth demands from users and exploit the gain of statistical multiplexing, dynamic bandwidth allocation (DBA) algorithms are normally used to provision the shared upstream bandwidth. More specifically, ONUs will report their buffer occupancy to OLT, which will then allocate the upstream bandwidth to ONUs based on their bandwidth demands and their Service Level Agreement (SLA).

Some standards have been developed for PONs by both EFM of IEEE (EPON) and FSAN of ITU-T (GPON). EPON is designed for carrying Ethernet frames and GPON can carry various traffic, such as Ethernet frames and ATM cells, through encapsulation. Although EPON and GPON have different frame structures, they share the same network architecture and employs the same data communication principles. One important difference between EPON and GPON is that GPON provides granular QoS definitions, while EPON requires adherence to the simple Multi Point Control Protocol (MPCP); hence GPON is equipped with a well-defined QoS framework, a feature highly preferred by the Internet Service Providers (ISP) for comprehensive traffic management in the access and last-mile networks. XG-PON standard, the successor of GPON and the prime focus of our work, is given in Section 'XG-PON details', in detail.

ns-3 network simulator

ns-3[1] is a state of the art open-source network simulator. Based on many lessons from the well-known ns-2 simulator[23], ns-3 is written from the scratch to be a completely new network simulator, with no backward-compatibility with the former. It is a discrete-event network simulator with the simulation core and modules of ns-3 implemented in C++;it is built as a library which may be statically or dynamically linked to a C++ main program. ns-3 also exports nearly all of its API to Python, allowing Python programs to import an ns3 module in much the same way as the ns-3 library is linked by executables in C++.

ns-3 has many attractive features, such as high emphasis on conformance to real networks, good support for virtualisation and testbeds, a novel attribute system for configuring simulation parameters, automatic memory management, a configurable tracing system and the ability to incorporate existing open-source modules through simple APIs[13]. It has also been reported that ns-3 performs much better than other simulators in terms of simulation speed and memory overhead[32]. The first release of ns-3 was made in

June 2008 with support for a number of modules including CSMA, Point-to-Point, WiFi (IEEE 802.11), TCP, UDP and IPv4.

In the last few years, many new modules have been developed and added into ns-3, such as WiMAX module from Inria[8] and LTE module from CTTC[27]. Thus, implementing an XG-PON module in ns-3 offers an established research platform for studying the issues that my occur in a deployment of XG-PON itself as well as in interfacing XG-PON with other network technologies.

Related work

Although simulation has been used to study PON in the past, such work cannot be used directly or extended easily to study the performance issues arising with the deployment of XG-PON. Song H, et al. developed their own simulator to study dynamic bandwidth assignment (DBA) algorithms when the physical reach is much longer than the current PON networks[30]. This simulator has limited functions and there is no Internet protocol (IP) stack, which is needed to study many research topics.

EPON, GPON and XG-PON[4;7;9;24;26] have all been implemented in OPNET[31]. However, these models tend to be abstracted away from the relevant standards definitions, and thus do not represent issues that may occur at lower layers - particularly the Physical (PHY) and Medium Access Control (MAC) layers - of a large-scale XG-PON deployment. They were also implemented at bandwidths lower than 1 Gb/s. Finally, as OPNET is not an open-source simulator, there is limited public access to the models, and we cannot adapt the OPNET's core to simulate a 10Gb/s XG-PON network with a reasonable simulation speed.

On the other hand, a simple EPON module has been developed for OMNeT++[6] and the code is available to the public, thereby providing a better simulation platform to understand communication in and via a PON network with conformance for standardisation. However, due to the differences between EPON and GPON discussed in 'Background' section, this EPON code will not be very helpful to implement a GPON or XG-PON module for OMNeT++, so that communication through (X)G-PON can be studied in detail.

Summary

Hence, there is a requirement to design and implement a standard-compliant XG-PON module from scratch for the state-of-the-art and open-source ns-3 simulator. With such an XG-PON module, we can simulate XG-PON at its actual data rates with reasonably swift simulation speeds; such a scenario would pave the way for researchers to identify and validate issues that may occur in a large scale XG-PON deployment. With the more realistic IP stack of ns-3, we can also study the performance experienced by users/applications in XG-PON networks. With the existing ns-3 modules for various wireless networks (WiFi, WiMAX, LTE, etc.), we can study the integration between XG-PON and wireless networks, which is the trend of the future Internet access networks.

Besides XG-PON, we can also extend this XG-PON module to study Long-Reach PON (LR-PON), an evolution

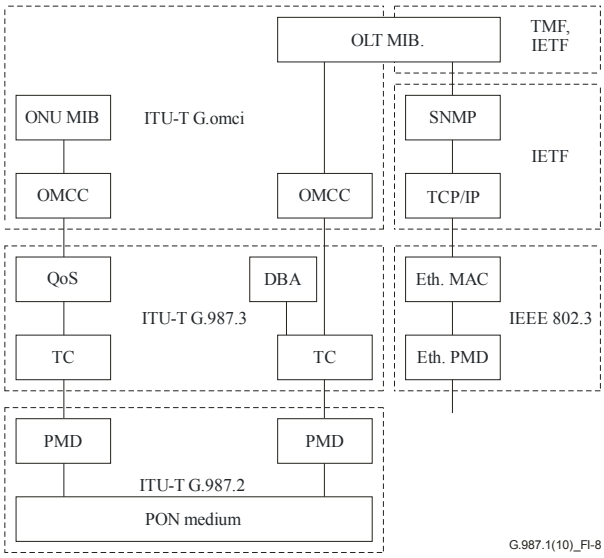

Figure 2. XG-PON common functions[18]

of XG-PON with a larger number of users, symmetric data rate (10 Gb/s in both upstream and downstream), and longer reach (100+ km)[25;29]. The aim of our LR-PON research group is to initially build LR-PON from the XG-PON standard, while identifying the required modifications and improvements.

The initial design and implementation of our XG-PON module had been reported in the 4th Workshop on ns-3[33]. Since then this module has been significantly redesigned and extensively evaluated. In this paper, we will discuss the design principles of the latest XG-PON module, describe its implementation details and present an extensive evaluation of the same.

XG-PON details

The XG-PON standard has many similarities with GPON, such as its TDMA scheme used to share the medium, the mechanism to provide QoS, and the DBA scheme used for the upstream bandwidth assignment. However, some changes are required in order to support a larger number of users, higher data rate, and extended physical reach. In this section, we will present the details of XG-PON.

Overview of XG-PON

A series of recommendations has been released by FSAN of ITU-T for XG-PON. ITU-T G.987 explains several important concepts of XG-PON; ITU-T G.987.1 presents the general requirements, services to be supported, hardware specifications and protocol stack of XG-PON as well as the network migration from and coexistence of XG-PON with GPON; ITU-T G.987.2 focuses on issues of the physical media dependent (PMD) layer, such as the used wavelength and the supported data rates whereas ITU-T G.987.3 presents the details of transmission convergence (TC) layer for XG-PON. Besides the protocols for data communication, it also covers QoS management and Dynamic Bandwidth Assignment (DBA) scheme for the upstream wavelength. Another related recommendation is ITU-T G.988, which specifies ONU management and control interface (OMCI)

for both GPON and XG-PON. Figure 2 illustrates XG-PON common functions and the recommendations in which they are specified.

Network architecture

XG-PON has been proposed for various deployment scenarios to serve different customers, such as residential, business, and cell site. To serve these customers, XG-PON lists the services to be provided, such as Telephony, high speed Internet access, mobile backhaul, etc. XG-PON also introduces many ONU variants that provide different functions and interfaces. In summary, XG-PON has been well standardized for providing full services to various users using optical network. As for optical distribution network, XG-PON can be deployed as a classical PON with its reach up to 60km. To support longer physical reach, active reach extenders (RE) can be applied before and/or after the passive splitter to connect multiple passive segments belonging to a single XG-PON network. These REs can be optical amplifiers or optical- electrical-optical regenerators that could fulfil the necessary optical link budget.

PMD Layer

There are two flavours of XG-PONs based on the upstream line rate: XG-PON1, featuring 2.5 Gb/s and XG-PON2, featuring 10 Gb/s in the upstream. The downstream line rate is 10 Gb/s in both XG-PON1 and XG-PON2. ITU-T G.987.2 focuses on the PMD layer for XG-PON1. XG-PON2 has not been standardized yet. In XG-PON1, the used wavelengths are 1575-1580nm (downstream) and 1260-1280nm (upstream). The exact downstream line rate is 9.95328 Gb/s and the upstream one is 2.48832 Gb/s. For line coding, NRZ (Non-Return to Zero) is used for both directions. ITU-T G.987.2 also specifies the requirements for hardware, such as optical fibre, transmitter/receiver, etc.

Transmission Convergence Layer

The XG-PON Transmission Convergence (XGTC) layer contains the definition for the MAC protocol of XG-PON. The XGTC layer maintains logical connections between OLT and each ONU, in pairs, in order to carry one downstream and one upstream traffic between the OLT and a corresponding ONU. Each connection is identified by a unique XG-PON encapsulation method (XGEM) Port-Id, which, while ensuring that packets are sent to the correct ONU, associates every connection to a certain Quality of Service (QoS) agreement. Note that one connection can only be configured to carry either a downstream or upstream traffic. To reduce the overhead of the DBA scheme, upstream bandwidth is allocated to groups of connections belonging to a single ONU. These groups are designated as Transmission Containers (T-CONT) and each group/T-CONT is identified by a unique identifier, the Alloc-Id.

XGTC comprises of three sublayers, with service adaptation sublayer on the top of the protocol stack, followed by framing sublayer and PHY adaptation sublayer.

Service Adaptation Sublayer: The service adaptation sublayer is responsible to adapt the upper layer traffic to the transmission mechanisms of XG-PON. It does this by mapping upper layer traffic to the corresponding connections, encapsulating/decapsulating data, segmenting/reassembling service data units (SDU) when necessary and inserting padding when there is insufficient data to fill an XGTC frame. If needed, it is also this sublayer's responsibility to encrypt/decrypt SDUs.

When mapping upper layer data to and from the connections of XGTC layer, while the OLT will maintain information pertaining to all the connections, an ONU will only maintain the connections that it owns. When the upper layer has something to transmit, it is also the service adaptation sublayer's responsibility to select the connections to be served according to their QoS parameters. When a connection is scheduled to be served, the service adaptation sublayer will then get data from its queue and insert an XGEM header to create an XGEM frame. The XGEM header will contain an XGEM Port-Id and some other information related to segmentation, padding, encryption, etc. When receiving an XGEM frame, the service adaptation sublayer will get the XGEM Port-Id from the XGEM header. If the corresponding connection exists in the connections maintained by the OLT/ONU, this sublayer will carry out reassembly (if necessary) and pass the data to upper layer. Otherwise, this XGEM frame will be discarded.

Framing Sublayer: In XG-PON, the OLT will send downstream XGTC frames every 125 μs, to broadcast traffic to all ONUs. In the upstream, ONUs send variable length XGTC bursts to the OLT for their upstream traffic. The length and start time of these upstream bursts are determined by the OLT through a DBA algorithm. The framing sublayer is responsible to generate and parse these XGTC frames/bursts. When generating a downstream XGTC frame at the OLT, the framing sublayer gets XGEM frames from service adaptation sublayer and joins them together into an XGTC payload. To create an upstream XGTC burst at ONU side, the framing sublayer may create multiple XGTC payloads, where each payload will carry XGEM frames from a single T-CONT. When parsing an XGTC frame/burst, the framing sublayer will send its payloads to the service adaptation sublayer for further processing.

In the header of the upstream XGTC burst generated by an ONU, there might be queue occupancy reports for the T-CONTs of this ONU. For each downstream XGTC frame, its header contains a BW_{map}, which instructs ONUs to share the upstream wavelength in a TDMA-like manner. More specifically, BW_{map} specifies the size of bandwidth allocations for T-CONTs, the used burst profile (the length of preamble, the length of delimiter, forward error correction or not, etc.), and the time to start to transmit. Since the OLT-ONU physical distance could be quite different for ONUs, each ONU should adjust the start time for avoiding collision in the upstream direction. Note that when an ONU is activated, the ranging procedure is carried out between the OLT and this ONU to determine how to adjust the start time of its upstream bursts.

Figure 3 illustrates the time-lines in XG-PON. The OLT and ONUs have a common view of the logical one-way delay of the optical distribution network (the largest one-way propagation delay plus various processing delays) and each

ONU uses its own equalization delay (EqD calculated in ranging procedure) to avoid collisions in upstream direction.

In the header of an upstream XGTC burst, the ONU can send one PLOAM (Physical Layer Operations, Administration and Maintenance) message to the OLT. As for a downstream XGTC frame, the OLT can send multiple PLOAM messages to multiple ONUs. Through exchanging PLOAM messages, many XGTC functions (key management, ONU power management etc.) can be fulfilled.

PHY Adaptation Sublayer: PHY adaptation sublayer interacts with PMD layer directly. Its main functions are forward error correction (FEC), scrambling, and frame delineation through a Physical Synchronization Block (PSB). In the downstream, the PHY adaptation sublayer will get an XGTC frame to create a PHY frame. These PHY frames are sent continuously every 125 μs. In the upstream, the PHY adaptation sublayer will get the XGTC burst and create a PHY burst. These PHY bursts have variable length due to the variable-length XGTC bursts. In the PHY burst, the PSB is determined by a burst profile selected by the OLT (through the BW$_{map}$) among the burst profiles, that are configured through PLOAM messages.

Scheduling and DBA

To decide the data to be transmitted in a downstream XGTC frame, a downstream scheduler is used by the OLT. Based on QoS parameters and service history of these downstream connections, the downstream scheduler will decide the connections to be served and the amount of data to be transmitted for each of them.

As for the upstream scheduling, the OLT uses a DBA algorithm to allocate the upstream bandwidth to T-CONTs. The DBA algorithm makes decisions based on queue occupancy reports, QoS parameters, and service history of these T-CONTs. The DBA algorithm needs to select the T-CONTs to be served, reserve a short gap time between the consecutive XGTC bursts for tolerating clock synchronization errors, determine the size of each bandwidth allocation, and calculate the start time for each bandwidth allocation. These decisions are broadcast to ONUs through BW$_{map}$. Since the upstream bandwidth is allocated to T-CONTs and each T-CONT may have multiple upstream connections, the ONU also needs an upstream scheduler to determine the upstream connections to be served during one transmission opportunity assigned to one T-CONT.

These scheduling algorithms, especially the DBA algorithm, are very important to network performance and QoS management. To allow competition and encourage research, these algorithms were intentionally left out of the standard. Indeed, it has been a very hot topic to study DBA algorithms for EPON and GPON [12;22;30]. Hence, there should be many research opportunities for XG-PON too.

Design principles and key decisions

When designing and implementing an XG-PON module, many issues must be considered and several trade-offs must be made when the goals conflict with each other. This section presents the design principles followed by the key decisions we made to realise a simulation-friendly XG-PON module in ns-3.

Design principles

Standard compliance: The ultimate goal of our research is to improve the performance issues associated with a deployed XG-PON network. Hence, when our simulation model has a close resemblance to a real world XG-PON deployment, we will be able to identify more realistic problems and provide solutions that can directly be applied in deployments. Hence, we will follow G.987 Recommendations from the FSAN group of ITU when designing our XG-PON module.

Simplicity: Considering that XG-PON is quite a complex standard, it will take a very long time to simulate a high-precision PON network, including all the layers, ranging from physical to network management. For instance, the document for ONU Management and Control Interface (G.988) is more than 500 pages. Hence, only the key components that are necessary to create a simple, yet working XG-PON network, are given priority in our module design; less important definitions that require extensive and complex implementation are added as stub classes for future extensions. For instance, since we are mainly interested in XGTC layer and upper layer issues, we can simulate the physical layer in a very simple way. We can assume that power budget for the optical distribution network has been satisfied through various techniques. The reach extenders and passive optical splitters/jointers need not be simulated. The channel, that simulates the optical distribution network of XG-PON, can simply pass downstream frames to all ONUs and pass upstream bursts to the OLT. As for Forward Error Correction (FEC), instead of the algorithm itself, we can simulate only its effect, i.e., the bandwidth overhead and the much lower packet corruption rate. Further details of omissions of less important features are explained in 'XG-PON Details' section, as we describe our module in detail.

Extensibility: When designing the XG-PON module for ns-3, we should also consider its extensibility since many other research topics might also be studied using this module. Hence, the extensibility is very important. When designing the class architecture of the XG-PON module, we followed the standard principles of Object Oriented Programming (OOP) such as using abstract classes to design easily extensible interfaces. Such abstraction, while providing full implementation for the key components of XG-PON, also pave way for quick integration of future extensions. For instance, when designing the class interface for the channel that simulates the optical distribution network of XG-PON, we should enable researchers to specify the tree structure of fibres, reach extenders, and splitters. When adding an ONU, they can also specify the splitter that it will be attached to and the physical distance between them. With this interface, it is possible to simulate the optical signal propagation and the possible packet corruption. However, for the current phase, we can let the channel store a list of ONUs and pass the downstream frames to all of them (without any error)[3]. Since DBA is a hot research topic, the classes for DBA should be well designed to allow the easy implementation of various DBA algorithms.

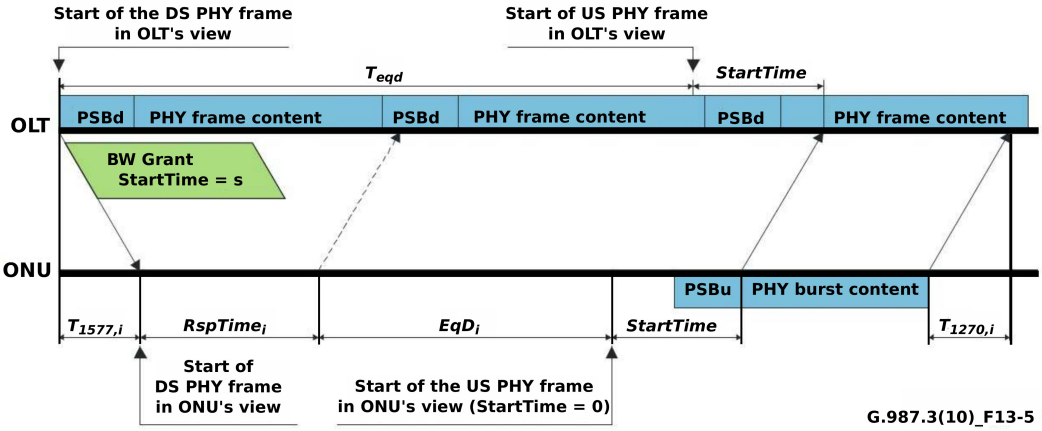

Figure 3. Time-line in XG-PON[18]

Configurability: In a single simulated XG-PON network, there could be thousands of nodes, such as the OLT, hundreds of ONUs, and hundreds of data traffic generators/sinks in core networks. Many nodes will also be attached to ONUs through various networks and act as traffic generators/sinks. Thus, while exporting as many configurable parameters as possible, we should also provide default parameters for most of them. Other methods, such as using helpers, should also be considered to ease the researcher's task of configuring the XG-PON network for simulation.

Simulation speed: Since the XG-PON to be simulated is a 10Gbps network, simulation speed must be considered at all times. A module, that can simulate XG-PON accurately (but very slowly), is useless for our research in which extensive simulations are needed. Saving CPU cycles and memory are primary focus here. For instance, when XG-PON is fully loaded in downstream and the size of each packet is 1KBytes, the simulator needs to process around one million packets per second. Since XG-PON could have hundreds of ONUs, the simulator must run the procedure used by ONU to judge whether it is the destination of one XGEM frame one billion times per second. This procedure must be implemented with high-efficiency. Straightforwardly, we can add one vector at each ONU whose index is XGEM Port-Id. When configuring XGEM Ports for this ONU, this vector can be marked correspondingly. Consequently, this vector can be used to filter out the traffic for this ONU quickly. However, XGEM Port-Id is a 16-bit number and this vector can consume a lot of memory when the number of ONUs is large. Hash map, in which GEM Port-Id acts as the key, suffers from the same memory issue. In our XG-PON module, we impose some simple relationship among XGEM Port-Id, Alloc-Id, ONU-ID, and IP address of the computer that this XGEM Port belongs to. Consequently, we consume a small amount of memory in total and achieve $O(1)$ time complexity when mapping IP address/XGEM Port-Id to the corresponding data structure.

During the implementation, many useful features of C++ language should be exploited and black-holes in CPU cycles must be avoided. First, we should pass parameters by reference whenever it is possible; *const reference* is preferred. We should also know that the smart pointer provided by ns-3 is fundamentally an object, though small. When the function is called frequently and some of its parameters are smart pointers, we should replace them with the reference of that smart pointer. Second, since C++ allows one class to override its *new* and *delete* operators, we should exploit this feature for data structures that are created and destroyed dynamically and frequently. By overriding these two operators, a pool of pre-allocated memory is used for small and dynamic memory requirements, thereby avoiding frequent calls to *malloc*/*free* and saving CPU cycles as a result. Third, when we select the data structure for a sequence of objects, *vector* should be considered due to its efficiency. However, when too many objects are added into one vector, reallocation may occur and the simulation can be slowed down significantly. Thus, we should reserve enough memory if the largest vector size can be pre-determined. Otherwise, *dequeue* should be considered as the container.

Additionally, virtual functions and inheritance are very attractive mechanisms in OOP that is exploited in designing classes for our XG-PON module. However, depending on the behaviour of entities in XG-PON module, there are instances when these mechanisms would result in cumbersome parent classes, requiring the use of downcast instead. Downcast, though, is a very unsafe mechanism that consumes a lot of CPU cycles at code execution. At such instances, we opted for designing classes individually, that is, without using inheritance, in order to expedite the overall simulation speed. For example, for each function of XGPON (DBA, etc.), there should be two classes designed for OLT and ONU, respectively, and it is attractive to let them inherit from the same parent. However, since the logic at OLT is totally different than that at ONU, the amount of reused code is limited, the interface of the parent becomes more complex, and simulation speed is slowed down. Thus, these classes are designed independently and the inheritance is not used.

Key decisions

Below are several key decisions we made, when designing and implementing our XG-PON module in ns-3. Corresponding implementation details will be presented in section 'XG-PON module for ns-3'.

Stand-alone simulation: Since XG-PON is a 10Gbps network with hundreds of ONUs, it would be very

attractive to use distributed simulation to speed up XG-PON simulation. Yet, it is necessary to have a robust cluster or multi-core computer to realise the advantages of MPI for a high-speed network such as XG-PON. On the other hand, although ns-3 supports distributed simulation through standard Message Passing Interface (MPI), this feature only works for point-to-point links. But XG-PON is fundamentally a point-to-multipoint network and extensive works are necessary to enable distributed simulation. These works include verifying MPI synchronisation against the simulation speed of XG-PON, obtaining reasonable speed in simulating an XG-PON with the added overhead of (de)packetization at cluster node interfaces, creating the topology of XG-PON in each core in a cluster as per MPI requirement and allocating ONUs to different cores. As we would rather prefer an XG-PON module which is both simple and used extensively among the research community, in this phase, we decided to model our XG-PON module as a stand-alone simulator, without support for MPI. That is our XG-PON module uses only one core even when one computer has multiple processors or cores. In the future, distributed simulation may be considered.

Packet-level simulation: Since ns-3 is an event-driven simulator, we have the options of simulating our XG-PON module at byte, packet or flow level. However, due to the high bandwidth of XG-PON (10Gbps) and the moderate frequency of state-of-the-art processors (mere several GHz), it is not feasible to simulate the extensive details of data transfer in XG-PON at byte (or bit) level. On the other hand it's too complex to model both XG-PON and TCP/IP protocol stack in a flow-level simulation with the added issue of not being able to study the potential subtle interactions between TCP/IP and XG-PON. Thus, we decided to simulate our XG-PON module at packet-level, to simulate as many details of data transfer as possible, while allowing sufficient interaction between TCP/IP and XG-PON layers. Furthermore, when passing traffic between OLT and ONU, all XGEM frames in the downstream frame or upstream burst should be handled together; as a result the number of simulation events can be reduced significantly. Due to the short XG-PON frame size (125μs), the upper layer protocols won't be affected if we keep the order of XGEM frames in the downstream frame or the upstream burst. Based on this decision, many physical layer operations, such as line coding and Forward Error Correction, will not be implemented in this module. However, the bandwidth overhead of FEC must be considered. Payload encryption/decryption will not be implemented as well, though the logic used for key management will be implemented for future extensions. We also assume that all the sub-modules in our XG-PON module complete their execution at the required point in time within every downstream/upstream frame, regardless of the complexity of the sub-modules. The total run time of the simulation however depends on the scale and complexity of the configured XG-PON network.

XG-PON in operation: Since we are mainly interested in the performance issues of an XG-PON network in operation, many aspects of XG-PON can be simplified. For instance, the activation procedure that uses PLOAM messages to add each ONU to an operational XG-PON need not be implemented. Instead, we can simply add all ONUs to the network before starting the simulation through a helper class. The ranging procedure that uses PLOAM messages to measure the one-way propagation delay of each ONU, can be simplified by setting the same delay values to both at the OLT and at the corresponding ONU at the time of configuring (before running the simulation of) the XG-PON network. In XG-PON, XGEM Port and T-CONT configuration is carried out through OMCI (ONU Management and Control Interface: G.988). For simplifying the dynamic configuration of XGEM Port and T-CONT by OMCI, we will configure all XGEM Ports and T-CONTs before starting the simulation through a helper class. Relevant stub classes will be designed in future extensions, for a detailed implementation of PLOAM and OMCI channels.

Simple Optical Distribution Network (ODN) and reliable data transfer: In XG-PON, the optical distribution network is quite complex and is comprised of many optical fibres, splitters/jointers, and reach extenders. However, we model the optical distribution network as a simple channel and we only simulate the propagation delay and line rates. We assume that the link power budget has been ensured through various techniques (reach extenders, etc.) and the laser receiver can work well. Thus, we will not simulate optical signal propagation (wavelength-dependent) and assume that all downstream frames and upstream bursts can arrive to their recipients correctly. In other words, transmission errors are not simulated in our XG-PON module. This is reasonable since FEC is normally applied to rectify transmission errors. Based on this decision, Cyclic Redundancy Check (CRC) and Header Error Correction (HEC) are not executed in the simulation.

In the future, transmission errors may be simulated at the recipient, by dropping an entire downstream frame or upstream burst with a distance-dependent probability. That is, an occurrence of transmission error, with no frame delineation, should be able to prevent a recipient from decoding an entire frame/burst when FEC fails to identify the frame/burst with sufficient accuracy.

Serialization avoidance and meta-data in data structures: Since this XG-PON module is designed for stand-alone simulation,(de)serialization is unnecessary and should be avoided[4]. Thus we defined our own data structure in this XG-PON module, instead of the Packet class from ns-3. This is because, though Packet supports easy insertion/extraction of headers, fragmentation and reassembly, when XGEM frame header is added into Packet, the header is serialized into one byte array. When one XGEM frame is received, the recipient needs to extract the XGEM frame header from the byte array, i.e., create a new data structure and carry out de-serialization. Considering that one XGEM frame in downstream direction will be processed by hundreds of ONUs, the above operations may consume too much CPU. To solve this issue, our data structure is designed to have one smart pointer to the header and another to the corresponding SDU (an instance of Packet). Hence, the header can be directly extracted from our data structure.

Another observation is that some meta-data can be added into data structures for various purposes since they are exchanged between OLT and ONU as objects (instead of a

byte array). For instance, all the broadcast XGEM frames in downstream need be checked by all ONUs, to decide whether to accept the relevant frames or to drop them. However, our observations indicate that the traffic in one downstream might belong only to a few ONUs, due to the small size of the downstream frames and the bursty nature of bandwidth allocation. Thus, to speed up the simulation significantly, we have added a bitmap to each downstream XGEM frame merely to indicate whether one ONU needs to check the particular frame, so that not every XGEM is checked by each ONU in detail for acceptance or dropping.

Extensible DBA, scheduling, and queue schemes: DBA engines, scheduling algorithms and the queue used by each XGEM Port at the sender side are very important to the performance of the whole network and the QoS experienced by user traffic. Thus corresponding classes were designed carefully to support future extensions. Creating abstract classes for these schemes, would allow new algorithms to be implemented easily, by redefining only a few key functions. Additional details of DBA classes implemented in our XG-PON module will be discussed in Section 'XG-PON module for ns-3', under 'Major modules'.

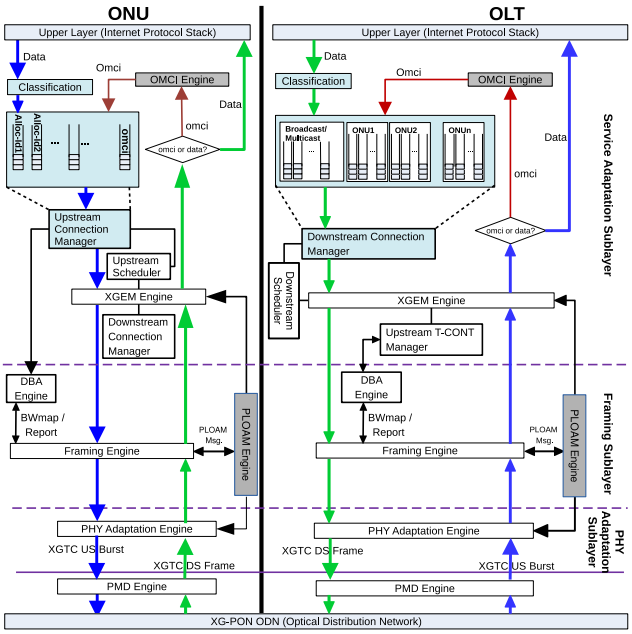

Figure 4. Functional block diagram of our XG-PON module

XG-PON module for ns-3

Our aim, in developing the XG-PON module for ns-3, is to provide a standard compliant, configurable, and extensible module that can simulate an XG-PON at reasonable simulation speed with support for a wide range of research topics. Hence, the below subsections explain the functional blocks as we modelled them in our XG-PON module, followed by its implementation details.

Functional blocks of our XG-PON module

Figure 4 explains the data transmission paths in both downstream (green arrow) and upstream (blue arrow) directions, as we modelled in our in our XG-PON module, based on the XG-PON standard [18] and our design principles.

Downstream traffic on OLT side: As shown by the green arrows in OLT side of Figure 4, when an SDU is received from the upper layer (eg: IP), the SDU should first be mapped to the corresponding connection (XGEM Port) based on the destination IP address and put into the queue for transmitting in the future. Thus, there must be an algorithm for mapping the IP address to an XGEM Port-Id.

Since the OLT needs to broadcast the downstream XGTC frames every 125 μs, it will periodically ask the OLT's *Framing Engine* to generate an XGTC frame. This engine will first generate an XGTC header since the available space for data in the frame depends on the size of the XGTC header. For the payload of a downstream XGTC frame, the Framing Engine resorts to the *XGEM Engine* to get an XGTC payload. This payload is comprised of concatenated XGEM frames that occupy all the available space. As for the SDUs to be encapsulated and transmitted, the XGEM Engine lets the *Downstream Scheduler* decide the connections to be served. This scheduler makes decisions based on *Downstream Connection Manager* which knows queue length, QoS parameters, and service history of each downstream connection. When carrying out encapsulation, fragmentation will be carried out by XGEM Engine if an SDU is too long for the current transmission opportunity. XGEM Engine is also responsible to encrypt these SDUs to avoid eavesdropping. The keys used for data encryption are negotiated through PLOAM messages and are maintained by *Ploam Engine*.

To construct the XGTC header of the frame, the *DBA Engine* is used to generate BW_{map} that tells ONUs how to share the upstream wavelength. DBA Engine makes decisions based on queue occupancy reports, QoS parameters, and service history of T-CONTs. PLOAM messages in the header, are generated by Ploam Engine. The downstream frame is sent to the ODN after passing through *PHY Adaptation Engine* and *PMD Engine*.

Downstream traffic on ONU side: Shown by the green arrows in ONU side of Figure 4, when a downstream PHY frame arrives at an ONU, it will pass through PMD Engine and PHY Adaptation Engine which will remove the physical-layer overhead. The Framing Engine is then responsible to parse the resulting downstream XGTC frame.

The PLOAM messages from the XGTC header will be given to the Ploam Engine, which will process the messages related with this ONU. The DBA Engine is responsible to process BW_{map} in the header, i.e., schedule its upstream XGTC bursts if required by this BW_{map}.

As for the payload, the XGEM frames are passed to XGEM Engine. Based on the list of its connections maintained by Downstream Connection Manager, the XGEM frames for this ONU are first extracted. XGEM Engine then carries out decapsulation, decryption, and reassembly (if needed)[5]. The received SDUs are then sent to the upper layer (eg: IP).

Upstream traffic on ONU side: As illustrated in Figure 4 by the blue arrows in ONU side, when a IP packet is received at the ONU, it is first mapped, based on the source IP address, to the corresponding upstream connection, which is maintained

by *Upstream Connection Manager*. The packet is then put into the corresponding queue for transmitting in the future.

When it is the time to transmit one upstream XGTC burst (as instructed in the BW_{map} sent by the OLT in a previous downstream frame), the Framing Engine in the ONU assumes responsibility for producing the XGTC burst. To do this, the *Framing Engine* in an ONU asks its *XGEM Engine* to get an array of XGTC payloads (SDUs), each of which is a concatenation of several XGEM frames belonging to an Alloc-Id. To decide the SDUs to be encapsulated, the *Upstream Scheduler* and the *DBA Engine* in the ONU are also needed since the upstream bandwidth is allocated to each Alloc-Id with the possibility of multiple upstream connections (or Alloc-Ids) belonging to the same ONU. Both the *Upstream Scheduler* and the *DBA Engine* make decisions based on several parameters such as the amount of bandwidth allocated to each Alloc-Id, queue length, QoS parameters, and service history of this T-CONT's upstream connections.

Framing Engine at an ONU also resorts to the *DBA Engine* to generate queue occupancy report for the corresponding Alloc-Id, when permitted by the OLT. This report is deduced by the *Upstream Connection Manager* based on the Alloc-Id(s) associated with each ONU. For various purposes, PLOAM messages may be generated by Ploam Engine. When it is allowed by the OLT, one PLOAM message can be put into the header of the XGTC burst, which may contain a range of XGTC SDUs multiplexed when the ONU has several upstream connections/T-CONTs.

The upstream XGTC burst is then passed to PHY Adaptation Engine with the burst profile to be used. After going through PMD Engine, this burst is sent to the ODN.

Upstream traffic on OLT side: As shown by the blue arrows in Figure 4, when the OLT receives an upstream XGTC burst from the XG-PON ODN, this burst first passes through *PMD Engine* and *PHY Adaptation Engine*. The *Framing Engine* at OLT is then responsible to parse the header and the payloads of this burst. The potential queue occupancy report will be sent to *DBA Engine* and the potential PLOAM message is sent to the *PLOAM Engine*. As for the XGTC payloads, they are sent to *XGEM Engine* for decapsulation and reassembly (if needed). Hence, an *Upstream T-CONT Manager* is needed to hold the potential segments for reassembly.

As illustrated in Figure 4, both OLT and ONU should have an *OMCI Engine* for exchanging OMCI messages that are used for various purposes (ONU management, XGEM Port and T-CONT configuration, etc.).

Implementation of our XG-PON module

Overview: Figure 5 illustrates a common simulation set-up that uses our XG-PON module and other ns-3 components to study the performance issues that may occur in XG-PON. The OLT is simulated as a node that has an *XgponOltNetDevice* and another network device, such as *PointToPointNetDevice*, to connect to an external network. The ONU is simulated as a node with an *XgponOnuNetDevice* and other network devices (Ethernet, WiFi, WiMAX, LTE, etc.) for connecting user equipments to the ONU. Thanks to ns-3, network devices of a node can be configured and we can study different deployment scenarios of XG-PON easily. Although XG-PON is proposed to carry

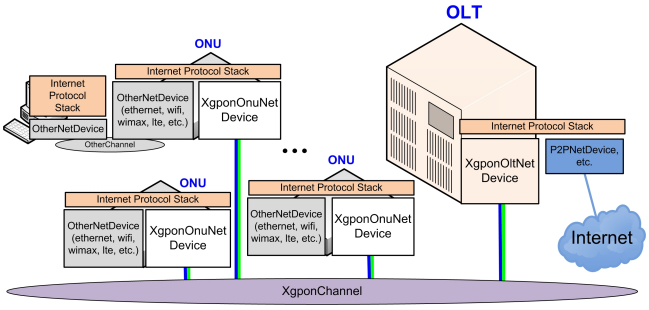

Figure 5. Sample XG-PON simulation environment

layer-2 frames of various network technologies (Ethernet, ATM, etc.), our XG-PON module interacts directly with the IP layer and IP packets are the SDUs. This is reasonable since we focus on FTTx networks connected to the Internet.

The OLT and ONUs are attached to *XgponChannel* that simulates the optical distribution network (ODN) of XG-PON. As illustrated in Figure 1, the ODN is a very complex tree composed by optical fibres, splitters/jointers, and REs. To produce trustworthy simulation results, it is highly desirable to simulate all details. But, XG-PON is also a very high-speed network with highly complex standard definitions. Hence, our module simplifies several aspects of XG-PON to reduce the development workload and increase the simulation speed.

Specifically, our *XgponChannel* merely simulates d_{max}, i.e., the logical one-way delay of the channel that is determined by the maximum propagation delay of ODN and various laser on/off delays at the ONU/OLT. d_{max} in our XG-PON module can be configured through the attribute system of ns-3. For a downstream frame from the OLT, *XgponChannel* will pass this frame to each ONU after waiting for d_{max}. *XgponChannel* passes the smart-pointer of this frame to each ONU, which will copy and process the data for itself [6] to avoid unnecessary data copy. As for an upstream PHY burst, *XgponChannel* will pass the corresponding smart-pointer to the OLT, after the appropriate d_{max}. EqD is calculated at the ONU when the upstream burst is produced, based on the BW_{map} from the OLT; EqD_i at ONU i is equivalent to the sum of d_{max} and the relative upstream burst delay (with regard to the previously scheduled ONUs in the same upstream frame) perceived at the XGTC layer by the *Framing Engine* in ONU i.

This means that although the difference of propagation delays among ONUs is simulated, the propagation of optical signals (fibre, splitter, etc.) is not simulated by the *XgponChannel*. It is also very CPU-intensive to calculate the optical signal strength for each downstream frame when it arrives at each ONU. Thus, our simplification is reasonable since the targeted research topics are related with MAC and upper layers. These simplifications also improve the simulation speeds significantly.

Major modules: Table 1 presents the summary of major classes used in our XG-PON module. We have also explained some significant sub-modules below, as they involve certain design choices. For information on all the classes our XG-PON module comprises of, the reader may refer to the documentation provided in sourceforge [2].

Table 1. Major classes in XG-PON module in ns-3.

Classes	Functionality
XgponChannel	Represents the physical medium of ODN
XgponNetDevice	Communicate with both upper layers and *PonChannel*, implements statistics related to OLT and ONU, defines various engines representing protocol stack of XG-PON
XgponDsFrame	Frame transmitted over XG-PON for downstream data
XgponUsBurst	Frame representing upstream burst
XgponXgemFrame	Represent XGEM frame and includes the payload and header
XgponXgemHeader	Represents the XGEM header defined in XG-PON standard
XgponTcont	Represents a T-CONT
XgponTcontOnu	Maintains queue occupancy reports from ONU, QoS parameters and service history of this T-CONT for DBA algorithm. Also holds the received segments for reassembly
XgponOnuConnManager	Contains a list of downstream connections and a list of T-CONTs in each ONU and implements both Downstream and Upstream Connection Managers for the ONU.
XgponOltConnManager	Downstream Connection Manager and Upstream T-CONT Manager for the OLT. Contains broadcast and uni-cast downstream connections and T-CONTs for upstream connections
XgponPhy	Implements PMD Engine and PHY layer parameters common to OLT and ONUs
XgponOltFramingEngine	Generate the downstream XGTC frames and parse the upstream XGTC bursts in OLT
XgponXgemRoutines	Implements routines common for both OLT and ONU (eg: XGEM frame creation) It also implements encapsulation, decapsulation, fragmentation, reassembly etc
XgponOltDsScheduler	Acts as the OLT Downstream Scheduler shown in Figure 4
XgponOltSimpleDsScheduler	A sub-class that follows the round robin (RR) scheme for downstream scheduling
XgponOnuUsScheduler	The ONU upstream scheduler shown in Figure 4. Also decides which connection in each ONU to be served in the next transmission opportunity, after payload generation
XgponOnuUsSchedulerRoundRobin	A subclass to serve the connections of every ONU in a round robin manner
XgponHelper	Helper class for configuring an XG-PON network
XgponConfigDb	Database that holds the information used by *XgponHelper*

Connection Managers: For both *XgponOnuConnManager* and *XgponOltConnManager*, we have implemented two subclasses in which these data structures are organized in different ways for different purposes: (1) *XgponOnuConnManagerSpeed* and *XgponOltConnManagerSpeed* impose some relationships among XGEM Port-Id,d Alloc-Id, ONU-ID, and IP address of the computer connected to ONU. They can carry out mapping very quickly, but they also limit the number of XGEM Ports that an ONU could have; (2) *XgponOnuConnManagerFlexible* and *XgponOltConnManagerFlexible* do not have such limitations,but they are much slower. Since millions of packets need to be processed per second, we recommend (1) for most cases.

PMD and PHY Adaptation: PMD Engine and PHY Adaptation Engine in Figure 4 are simplified significantly for simulating XG-PON with reasonable speed. The most important function of the interface here is to tell other classes about the size of an downstream/upstream frame. *XgponOltPhyAdapter* and *XgponOnuPhyAdapter* are used to implement PHY Adaptation Engine for the OLT and ONU, respectively. Instead of simulating their functions (line coding, FEC, scrambling, etc.) step by step, they just pass frames/bursts between *XgponChannel* and Framing Engine after removing physical layer header. Hence, we implicitly assume that all frames/bursts can be received correctly. Since the network should be well planned and FEC has been adopted, the observed frame corruption rate will be very low and this assumption is reasonable. In the future, the corruption rate of frames will be simulated based on the distance between OLT and ONU or empirical measurements of XG-PON networks in real world.

DBA: To study different scheduling and DBA schemes, several abstract classes are used in this module for extensibility. The actual schedulers can then inherit these abstractions and implement their specific algorithms. For example, *XgponOltDbaEngine* is designed for the OLT DBA Engine shown in Figure 4. When *XgponOltFramingEngine* generates one downstream XGTC frame, it will resort to *XgponOltDbaEngine* to generate a BW_{map}. *XgponOltDbaEngine* is also responsible to receive queue occupancy reports from ONUs. Currently, a simple DBA algorithm is implemented in *XgponOltDbaEngineRoundRobin*, which serves fixed amount of bytes for each T-CONT in a RR manner. Similarly, we could also implement the modified GigaPON Access Network (GIANT) DBA[22], which was initially proposed for GPON and the recent Efficient Bandwidth Utilization (EBU) DBA[11] in our XG-PON module, for supporting different classes of T-CONTs with multiple QoS parameters. *XgponOnuDbaEngine* acts as the ONU DBA Engine shown in Figure 4. It is responsible for processing BW_{map}, generating queue occupancy report, and scheduling upstream burst.

Helper: For facilitating researchers to configure an XG-PON network with hundreds of ONUs and thousands of connections, *XgponHelper* is also implemented in this module. Through XgponHelper, researchers can install *XgponNetDevice* on nodes and attach them to *XgponChannel*. They can also configure XGEM Ports and T-CONTs for carrying user traffic. Researchers can also use *XgponHelper* to enable Ascii and Pcap tracing.

Miscellaneous: Further classes of interest in our implementation are listed here.

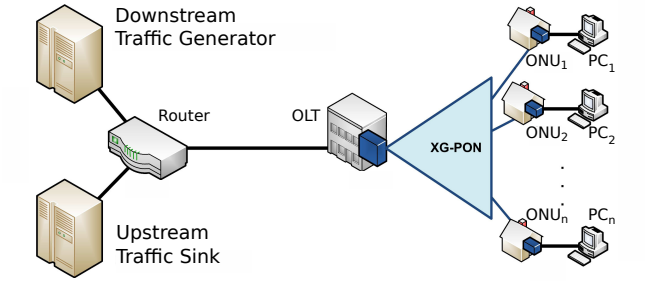

Figure 6. Simulated network topology

- As explained in subsection 'Key decisions', we used *XgponXgemFrame* to represent XGEM frame in our XG-PON module, instead of *Packet* class from ns-3
- *XgponOltPloamEngine* and *XgponOnuPloamEngine* are designed for exchanging PLOAM messages between the OLT and ONU. They also use *XgponLinkInfo* to maintain per-ONU information, such as keys and burst profiles.
- *XgponOltOmciEngine* and *XgponOnuOmciEngine*, are designed for implementing the OMCI channel. For these classes though, we have only implemented their interactions with other layers. We will simulate their messages and the related procedures in the future.
- *XgponOnuUsScheduler* is put within *XgponTcontOnu* so that T-CONTs of the same ONU may use different scheduling algorithms for their upstream traffic.
- *XgponConfigDb* uses one flag to make sure that *XgponOltConnManagerSpeed*, *XgponOnuConnManagerSpeed*, and *XgponIdAllocatorSpeed* are used together.

Evaluation results

In this section, with several typical simulation scenarios, we first demonstrate that our XG-PON module, designed for simulating a 10Gbps optical network with hundreds of ONUs, can indeed work as expected. Then we evaluate our XG-PON module's simulation speed and memory consumption under different load and ONU numbers using an off-the-shelf server, since simulation performance is one of the most important metrics in large-scale and high-speed network modelling. Extensive pressure tests are also carried out to demonstrate that our XG-PON module can run for a very long time and work as expected under controlled and random simulation environments.

Figure 6 shows the network topology used in our evaluations. We simulate an XG-PON network whose d_{max} is 0.4ms, which is more than the one way propagation delay in fibre (= 0.3ms) for a refractive index of 1.5 and a physical reach of 60km. For the data rates of XG-PON, we follow XG-PON1, which is capable of 10Gbps in downstream and 2.5Gbps in upstream. There is a total of n ONUs in the XG-PON and one PC is connected to each ONU through a point-to-point link. These PCs act as the customer of XG-PON and play the role of generators for upstream traffic and that of sinks for downstream traffic. Delay between each PC and an ONU is set at 2ms, indicating a maximum one-way delay between the user application and the ISP terminal near the user. The OLT is connected to a Router and the

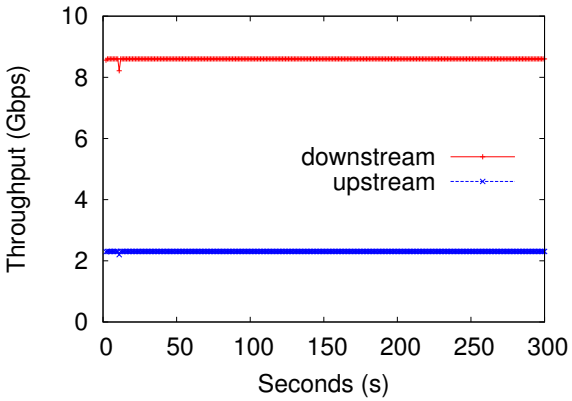

Figure 7. Effective bandwidth of XG-PON1

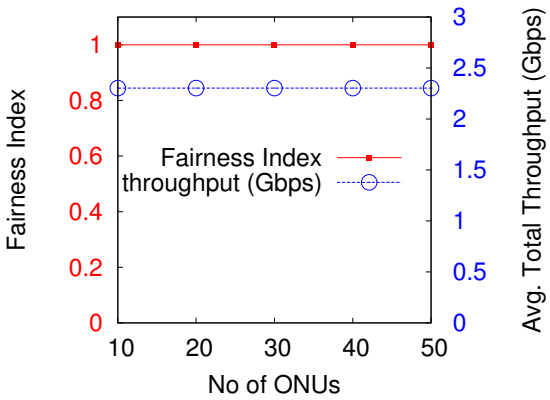

Figure 8. Fairness of Round-Robin-based DBA Algorithm

point-to-point link between them is used to simulate the core network. More specifically, the delay of this link is set to 10ms, which is a practical one-way delay between routers at ISP and an OLT placed at the ISP edge of an access network. Downstream Traffic Generator and Upstream Traffic Sink are connected to Router through point-to-point links whose delay is 2ms, which again is a realistic representation of a one-way delay between an application at the ISP and its core routers. The bandwidth of all the above point-to-point links is set to 20Gbps so that XG-PON is the only bottleneck link in the entire simulation environment. At the application layer in the Downstream Traffic Generator/PCs, we use traffic models with uniform inter packet arrival time distribution, to generate the user traffic. More specific details of the network traffic will be presented in the respective subsections.

Functionality validation

Effective bandwidth of XG-PON1: The data rates of XG-PONs are 10Gbps in downstream and 2.5Gbps in upstream. However, due to the overhead of FEC (Forward Error Correction) and the headers of various layers, the effective bandwidth observed by applications is much less. In this experiment, 256 ONUs are simulated, UDP traffic is generated in both directions, and the overall network load is maintained higher than the data rate of XG-PON in each direction.

Figure 7 plots the effective bandwidth observed by applications with time. The red line indicates that the overall throughput in downstream is around 8.5Gbps while the blue line indicates that the overall throughput in upstream is around 2.3Gbps; these data rates are equivalent to

theoretical values, given a packet size of 1024 Bytes with the above overheads are accounted for. Thus, these results show that our XG-PON module has properly simulated SDU encapsulation (XGEM and XGTC headers) and other overheads (FEC, inter-burst gap, etc.).

Fairness of Round-Robin-based DBA algorithm: DBA is one of the most complex functions of XG-PON and it is necessary to verify that it works as expected. In this group of experiments, we will demonstrate that our round-robin-based DBA algorithm can allocate the upstream bandwidth fairly and efficiently. In each experiment, n ONUs are simulated, customers (the PCs connected to ONUs) generate the same amount of upstream UDP traffic, and the overall network load in upstream is higher than the upstream data rate (=2.5Gbps) of XG-PON1; n is set to 10, 20, 30, 40, and 50.

Figure 8 plots Jain's fairness index [19] among the customers and the overall throughput in upstream. The red line shows that the fairness index is equal to 1, indicating that the round-robin-based DBA algorithm fairly allocates the upstream bandwidth to all the customers. The blue line indicates that our round-robin-based DBA algorithm is also efficient since the overall throughput is 2.3Gbps.

Trade-off between throughput and delay: The well-known trade-off between throughput and delay is also applicable to XG-PON, due to its large bandwidth-delay product (BDP). More specifically, when an ONU gets the chance to be served, the larger the maximal service size (MSS) used by the DBA algorithm is, the smaller the upstream burst overhead is and the higher the overall upstream throughput is. However, larger MSS also enforces larger interval between two consecutive services for the same ONU; as a result the packets wait longer at the ONU upstream buffer due to larger scheduling delay. In this group of experiments, 256 ONUs are simulated, customers generate the same amount of upstream UDP traffic, and the overall network load in upstream is higher than 2.5Gbps. The MSS is set to 500B, 1KB, 2KB, 4KB, 8KB, and 16KB. Validation is considered only for the upstream since the round-robin DBA is employed only for upstream bandwidth allocation.

Figure 9 illustrates the impact of MSS on throughput and delay; both the overall upstream throughput (blue line) and the scheduling delay (red line) increase with MSS, indicating the existence of the well-known trade-off between throughput and delay in large BDP networks.

TCP in downstream and upstream directions: Here, we demonstrate the behaviour of a common congestion control algorithm from a realistic TCP stack to validate the ability of our XG-PON module to work seamlessly with TCP. First, we successfully integrated our XG-PON module with a real-world TCP stack from Linux Kernel(version 2.6.26) packed in Network Simulation Cradle (NSC [20]). Using this integration, we generated a single CUBIC TCP [10;21] flow across our XG-PON module, both in downstream and upstream directions, individually. Figure 10 shows the throughput vs. time plots for a single TCP connection in the downstream (red line) and the upstream (blue line). When the data rate at the sender is higher than network bandwidth, packets are dropped and sending rate (or throughput) is reduced. We can observe that at each TCP epoch, both in downstream and upstream, throughput curves in Figure 10

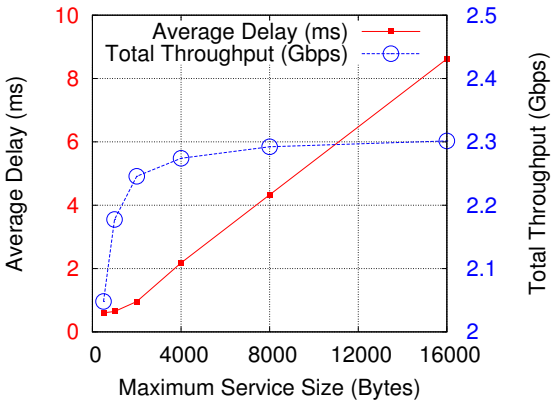

Figure 9. Effects of Maximal Service Size used by DBA

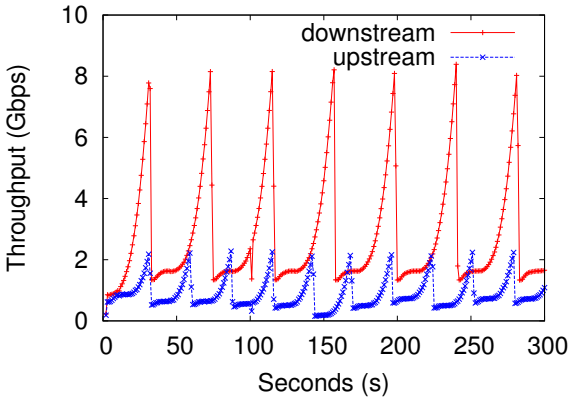

Figure 10. TCP traffic on XG-PON1 (throughput vs. time)

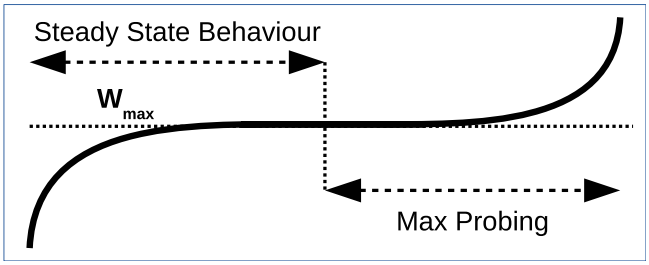

Figure 11. General CUBIC TCP congestion window growth [10]

matches well with the congestion window (cwnd) growth function of Cubic TCP (Figure 11) Steady State Behaviour and Max Probing periods; each CUBIC flow, at the end of every Max Probing period in Figure 10 shows heavy exponential growth of its cwnd until a congestion is caused by the downstream or upstream capacity XG-PON. We have also performed extensive simulations combining our XG-PON module and realistic TCP stacks (Linux Kernel from NSC) employing different congestion control algorithms [5] to validate that our XG-PON module is capable of accommodating real-world TCP stacks, under various scenarios.

Performance evaluation

In evaluating performance, one dedicated computer is used to measure performance of our XG-PON module, to avoid interference from other processes. We used Dell PowerEdge R320 rack server and the processor is Intel(R) Xeon(R) CPU E5-1410 0 @ 2.80GHz with a cache of 10MBytes. Note that although this processor has 4 cores, just one of them is used

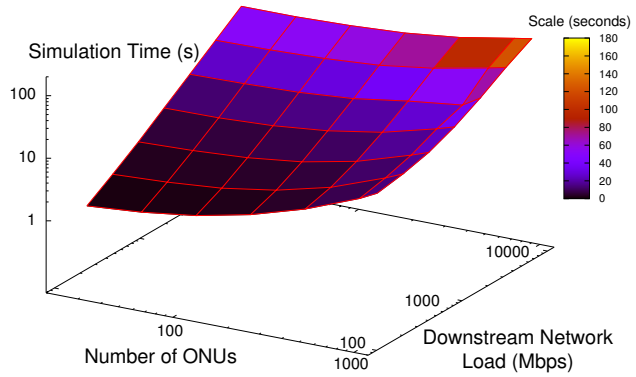

Figure 12. Simulation speed of our XG-PON module

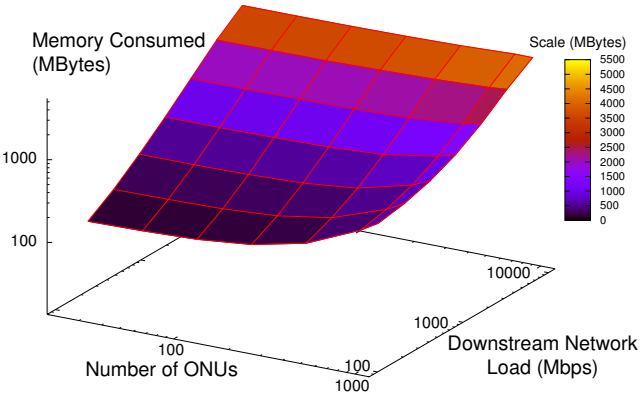

Figure 13. Memory consumption of our XG-PON Module

by our simulation. As for the main memory, the server is installed with 48GBytes in total.

To study the simulation performance of our XG-PON module under various scenarios, the number of ONUs and the amount of network traffic are changed in our experiments. The evaluated values of n (the number of ONUs) are 25, 50, 100, 200, 400, 800, and 1000; values for total amount of network load in the downstream are, 150Mbps, 300Mbps, 600Mbps, 1.2Gbps, 2.4Gbps, 4.8Gbps, and 9.6Gbps. Due to the overhead of XG-PON physical and XGTC layers, we observe packet drop when the downstream network load is beyond 9.6Gbps. Furthermore, for all experiments, the upstream network load is always one quarter of the downstream network load. Thus, when the downstream network load is 9.6Gbps, the upstream network load is 2.4Gbps and with observable packet drop in the upstream direction.

To evaluate the speed of our XG-PON module, 400 seconds are simulated in each experiment, the total amount of time used to complete the simulation is recorded; finally the average time consumed for each simulation second in each scenario is calculated and plotted in Figure 12. The results indicate that the average time consumed (shown by the vertical axis and the scale in the figure) increases linearly with the network load. This is reasonable since ns-3 is a packet-level network simulator and the number of events increase linearly with the number of packets. We observe, on the other hand, the consumed time increases at a slower pace with the number of ONUs. Figure 12 also indicates that our XG-PON module takes around 160s to simulate one second even when there are 1000 ONUs and the downstream network load is 9.6Gbps with which XG-PON is overloaded.

We also used *gdb*, the standard debugger for GNU software system, to run the simulation of the most difficult scenario (ONUs: 1000; Network load: 9.6Gbps) in *debug* mode. After the simulation enters into steady phase, we break it at random time, check the call stack, and continue the simulation. We observe that our XG-PON code occupies the CPU cycles only at 4 instances of the 126 such breaks. This clearly indicates that our XG-PON module is not the bottleneck of simulation speed and calls for additional investigation of other associated ns-3 modules (routing, etc.) to further improve simulation speed of our XG-PON module.

Besides simulation speed, we also evaluated the memory consumption of XG-PON simulation. The same experiments are repeated to collect these results. For each experiment, after starting the simulation, we wait for sufficiently long time until the simulation enters into its steady phase, where the amount of consumed memory does not increase any more. The recorded values, plotted in Figure 13, indicate that the consumed memory (again, shown by the vertical axis and the scale in the figure) increases linearly with the network load while it increases at a smaller pace with the number of ONUs. When there are 1000 ONUs and the downstream network load is 9.6Gbps, the consumed memory is still less than 5GBytes.

Based on the above results, we can conclude that with the off-the-shelf server, our XG-PON module can simulate an XG-PON network of 1000 ONUs and 10Gbps (downstream), with reasonable speed and moderate memory consumption.

Pressure tests

To evaluate the robustness of our XG-PON module, we carried out more experiments. We use the same configurations designed for performance evaluation and 4000 seconds are simulated in each experiment to demonstrate that our XG-PON module can run for sufficiently long periods (longer than one hour). Note that when there are 1000 ONUs and the downstream network load is 9.6Gbps, it takes more than one week to complete the simulation. In another set of 49 experiments, we randomly select the number of ONUs and the amount of network traffic, and 500 seconds are simulated in each experiment. All these simulations ran successfully without any interruption or memory leakage during the course.

In summary, these evaluation results indicate that our XG-PON module is sufficiently robust and can simulate XG-PON accurately with reasonable speed and moderate memory consumption. Thus it can be used as a very good research platform for studying performance issues related with XG-PON.

Summary and future work

In this paper, we introduced an XG-PON module for the ns-3 network simulator. We described the details of its design and implementation, and presented the evaluation results on both functionality and performance. The results indicate that our XG-PON module is quite robust and can simulate XG-PON correctly with reasonable speed and moderate memory consumption. As the first and a full-fledged XG-PON module for ns-3, we believe that this work is a significant contribution to the scientific community; for any interested researcher, our XG-PON module provides the opportunity to study the performance issues associated with the deployment of XG-PON, using a validated simulation module.

In the future, we will implement more scheduling and DBA algorithms proposed for GPON or XG-PON, keep improving the simulation speed, add support for parallel/distributed simulation and investigate how to simulate Fibre-to-the-Cell using this XG-PON module and WiMAX/LTE modules in ns-3.

Acknowledgements

This work is supported in part by the CTVR Grant (SFI 10/CE/I 1853) from Science Foundation Ireland

Notes

1. http://www.ctvr.ie
2. http://www.ucc.ie/en/misl/
3. Depending on the situation, it may be worthwhile to simulate a likely low packet corruption rate, with the effects of FEC
4. All data structures must provide one function to return its serialized size to accurately compose downstream frame and upstream burst.
5. For each downstream connection, the Downstream Connection Manager at ONU should hold the initially received segments for reassembly while the remaining segments are received.
6. In the future, this will be revised to support parallel simulation, where ONUs will be simulated in different CPUs of a cluster.

References

1. (2011-15) The NS-3 network simulator. Available at: http://www.nsnam.org.
2. (2014) XG-PON Simulation Module for NS-3. Available at:http://sourceforge.net/projects/xgpon4ns3/.
3. 3GPP (2008) Evolved Universal Terrestrial Radio Access (E-UTRA).
4. Anthapadmanabhan NP, Dinh N, Walid A and van Wijngaarden AJ (2013) Analysis of a probing-based cyclic sleep mechanism for passive optical networks. In: *2013 IEEE Glob. Commun. Conf.* IEEE. ISBN 978-1-4799-1353-4, pp. 2543–2548.
5. Arokkiam JA, Wu X, Brown KN and Sreenan CJ (2014) Experimental evaluation of TCP performance over 10gb/s passive optical networks (XG-PON). In: *Globecom 2014 - SAC Access Networks and Systems*. Austin, USA, pp. 2269–2274.
6. Bodozoglou A (2010) EPON for OMNeT++.
7. Chang CH (2008) *Dynamic Bandwidth Allocation MAC Protocols for Gigabit-capable Passive Optical Networks*. PhD Thesis, University of Hertfordshire.
8. Farooq J and Turletti T (2009) An IEEE 802.16 WiMAX Module for the NS-3 Simulator. In: *SIMUTools*.
9. Fernando DNV, Milosavljevic M, Kourtessis P and Senior JM (2014) Cooperative cyclic sleep and doze mode selection for NG-PONs. In: *2014 16th Int. Conf. Transparent Opt. Networks*. IEEE. ISBN 978-1-4799-5601-2, pp. 1–4.
10. Ha S, Rhee I and Xu L (2008) Cubic: a new tcp-friendly high-speed tcp variant. *Operating Systems Review* 42: 64–74.
11. Han MS, Yoo H and Lee DS (2013) Development of Efficient Dynamic Bandwidth Allocation Algorithm for XGPON. *ETRI J.* 35(1): 18–26.
12. Han MS, Yoo H, Yoon BY, Kim B and Koh JS (2008) Efficient dynamic bandwidth allocation for FSAN-compliant GPON. *Journal of Optical Networking* 7(8): 783–795.
13. Henderson TR, Lacage M and Riley GF (2008) Network simulations with the ns-3 simulator. In: *Sigcomm (Demo)*.
14. IEEE (2004) 802.3ah: Ethernet in the First Mile.
15. IEEE (2004) IEEE Std. 802.16-2004, IEEE Standard for Local and Metropolitan Area Networks - Part 16: Air Interface for Fixed Broadband Wireless Access Systems.
16. Ikeda H and Kitayama K (2009) Dynamic Bandwidth Allocation With Adaptive Polling Cycle for Maximized TCP Throughput in 10G-EPON. *Journal of Lightwave Technology* 27(23): 5508–5516.
17. ITU (2008) Gigabit-Capable Passive Optical Networks (G-PON). Rec. G.984.x.
18. ITU (2010) 10-Gigabit-Capable Passive Optical Networks (XG-PON) Series of Recommendations. G.987.x.
19. Jain R, Chiu DMW and Hawe WR (1984) A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. DEC Research Report TR-301.
20. Jansen S and McGregor A (2005) Simulation with real world network stacks. In: *Winter Simulation Conference*.
21. Leith D, RNShorten and GMcCullagh (2007) Experimental evaluation of cubic-tcp. In: *PFLDnet Workshop*.
22. Leligoun HC, Linardakis C, Kanonakis K, Angelopoulos JD and Orphanoudakis T (2006) Efficient medium arbitration of FSAN-compliant GPONs. *International Journal of Communication Systems* 19: 603–617.
23. McCanne S and Floyd S (1997) The LBNL network simulator (NS-2). Http://www.isi.edu/nsnam/ns/.
24. Orphanoudakis TG, Kosmatos EA, Matrakidis C, Stavdas A and Leligou HC (2014) Hybrid resource reservation scheme for transparent integration of access and core optical transport networks. In: *2014 16th Int. Conf. Transparent Opt. Networks*. IEEE. ISBN 978-1-4799-5601-2, pp. 1–4.
25. Payne DB and Davey RP (2002) The future of fibre access systems? *BT Technology Journal* 20(4): 104–114.
26. Peng Z and Radcliffe P (2011) Modeling and Simulation of Ethernet Passive Optical Network (EPON) Experiment Platform based on OPNET Modeler. In: *ICCSN*.
27. Piro G, Baldo N and Miozzo M (2011) An LTE module for the ns-3 network simulator. In: *WNS-3 in conjunction with SIMUTools*.
28. Postel J (1981) Transmission Control Protocol - DARPA Internet Program Protocol Specification. RFC 793.
29. Shea DP and Mitchell JE (2007) A 10-gb/s 1024-way-split 100-km long-reach optical-access network. *Journal of Lightwave Technology* 25(3): 685–693.
30. Song H, Kim BW and Mukherjee B (2009) Multi-Thread Polling: A Dynamic Bandwidth Distribution Scheme in

Long-Reach PON. *IEEE Journal on Selected Areas in Communications* 27(2): 134.
31. Technology R (2015) OPNET Modeler.
32. Weingartner E, vom Lehn H and Wehrle K (2009) A performance comparison of recent network simulators. In: *ICC*.
33. Wu X, Brown KN, Sreenan CJ, Alvarez P, Ruffini M, Marchetti N, Payne D and Doyle L (2013) An XG-PON module for the NS-3 network simulator. In: *Workshop on NS-3 (held with SIMUTools)*.