On the Impact of Software Failures on Time Division Multiplexed Passive Optical Networks Dependability

Álvaro Fernández, Norvald Stol
Department of Telematics
Norwegian University of Science and Technology
Trondheim, Norway
alvarof@item.ntnu.no, norvald.stol@item.ntnu.no

***Abstract*—Passive Optical Networks (PONs) are widely regarded as the best suited technology for deploying broadband access networks. As new services emerge, the dependability of PONs has become critical as end users expect access networks to be highly reliable. Although PONs dependability regarding hardware failures has been extensively studied, very little attention has been drawn to software failures in PONs. Chiefly, this paper aims at performing an exhaustive analysis of the effect of software failures in PONs dependability and failure-related costs. Additionally, hardware failures are also included for the sake of completeness and comparison. By applying Duane's model for reliability growth to current literature results, the PON software dependability is estimated as a function of the testing phase duration. Then, a Markov cost model, accounting for both hardware and software failures is developed and solved by means of Monte Carlo simulations. Hence, the effect of software failures in PONs asymptotic availability, failure impact and dependability-related costs is detailed; revealing to be of utmost importance. Moreover, how the testing process duration affects these three parameters is also pinpointed.**

Keywords–Availability; failure impact; Operational Expenditures; Passive Optical Networks; software failures;

I. INTRODUCTION

Over the last years, the increase in bandwidth requirements demanded by end users has pushed operators into the deployment of broadband access networks. Amid other options, Passive Optical Networks (PONs) are considered to be the best suited technology for meeting such demands [1]. PONs not only provide high bandwidth on a per-user basis, but are also scalable and flexible. Additionally, PONs present a relatively low-cost deployment and energy consumption when compared with other alternatives. Consequently, PONs have been already widely deployed, while Next-Generation PONs (NG-PONs) are regarded as the most promising solution for future broadband fiber-based access networks [2].

Yet, as time passes, end users and services also demand higher service dependability in addition to higher bandwidth. Telemedicine, interactive gaming or e-commerce have caused end users (both residential and business) to expect reliable service delivery. Subsequently, the importance and significance of access networks' dependability has arisen as a cause of concern over the past years. As a matter of fact, several protection schemes and dependability-cost analyses for different PON and NG-PON flavors can be found in literature [3], [4], [5], [6].

Generally, the dependability of a system is assessed by its asymptotic availability, a parameter which is close to the users' perception. Operators are also concerned about the number of clients affected by a failure (i.e. failure impact), as large outages can cause negative publicity. Besides, operators' interest in dependability is also focused on the costs associated to failures, which are part of the Operational Expenditures (OPEX). Dependability-related OPEX include the cost of repair, the payment of penalties and loss of reputation, especially if the failure impact or the outage times are large.

Still, most of the already published PON dependability studies are focused on hardware, physical faults and/or environment failures. Even though software faults are the cause of an important part (usually bigger than hardware faults) of service failures in many systems [7], [8]; very few papers address software dependability in PONs.

Hence, the aim of this paper is to give a deep insight into the effect of software failures in Time Division Multiplexed (TDM) PONs' dependability and failure-related OPEX. Based on the results in [9], where software bugs in Gigabit-capable TDM PONs (GPONs) were studied; this paper performs a thorough analysis of software failures in TDM PONs. By applying Duane's model for reliability growth [10] to these results, the failure intensity of software failures in TDM PONs is estimated as a function of the testing time. Then, a Markov cost model [11], including both hardware and software failures, is developed to capture the dynamic dependability behavior of PONs. Thus, this study is able to detail the effects not only of hardware failures, but also of software failures and software testing time in the availability, failure impact and dependability-related OPEX in TDM PONs.

The remainder of this paper is organized as follows. First, Sect. II introduces the basic PON architecture. Section III presents the software dependability modelling approach taken in this study. Section IV describes the Markov cost model employed to assess the dependability and dependability-related OPEX of hardware and software failures in TDM PONs. Section V presents the analysis results in terms of asymptotic availability, failure impact and dependability-related OPEX. Finally, Sect. VI gives the conclusions of this work.

II. PON ARCHITECTURE

In this section, the PON architecture assumed along the paper is presented.

Succinctly, the typical PON architecture presents a tree structure, as depicted in Fig. 1. At the operator's Central Office (CO), the Optical Line Terminal (OLT) is housed – the root of the tree structure. Two different elements are considered at the OLT: the OLT ports where fibers are connected and the OLT chassis that hosts the OLT ports. Resembling the leaves of the tree, the equipment at the user's side is denoted as Optical Network Unit (ONU). Amid the CO and the ONUs, the Remote Node (RN) is deployed and serves as splitting point. Similarly to the OLT, the RN consists of the RN chassis which accommodates the set of passive elements performing the signal splitting. Basically, the passive elements can be pure optical splitters for TDM PONs, Arrayed Waveguide Gratings (AWGs) for Wavelength Division Multiplexing (WDM) PONs, or a combination of both for Hybrid WDM/TDM PONs. As this study is focused on the impact of software failures in TDM PONs, GPONs in particular; splitters are assumed as passive elements at the RN. In accordance with the GPON ITU-T Standard [12], the splitters' split ratio is fixed to 1:32. Necessarily, the OLT equipment and software are also that of a GPON technology.

When regarding the fiber infrastructure, two different fiber sections can be identified. First, the fibers interconnecting the OLT and the RN, typically denoted as Feeder Fiber (FF). Generally, feeder fibers span over several kilometers as users in the same PON share the feeder fiber infrastructure. Second, the fibers laid between the RN and the final users, called Distribution Fibers (DF). Distribution fibers cover a smaller distance than feeder fibers, being 20 kilometers the maximum reach of the basic GPON technology [12].

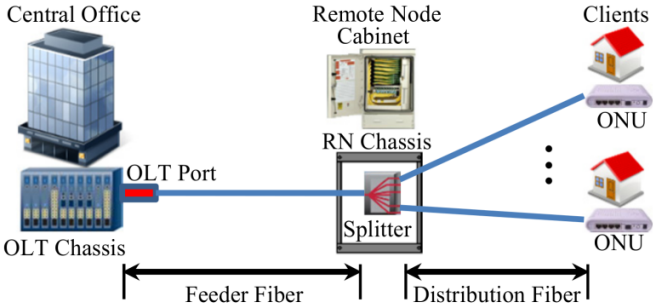

Fig. 1. Schematic PON architecture.

III. SOFTWARE DEPENDABILITY MODELLING

In this section, the model used to assess the dependability of the software in the OLT is introduced.

A. *Duane's Model for Software Reliability Growth*

Mainly, software reliability growth models have been developed in order to forecast the dependability of a software system. Reliability growth models measure the improvement of software reliability through the testing phase, typically predicting the software failure intensity at the end of the testing and debugging process [10]. As a result, the failure intensity (thus dependability) of the delivered software can be predicted.

Inversely, the dependability requirement can be fixed beforehand, and the reliability growth model employed to predict the time (and effort) necessary to meet this requirement. Among a vast number of reliability growth models, Duane's model has been chosen due to its simplicity and straightforward application. Intentionally, a brief description of Duane's model is presented here, while a more thorough description can be found in, e.g., [10].

Essentially, Duane's model is based on the observation that if the cumulative number of failures (N(t)) versus the cumulative testing time (t) was plotted on a log-log scale; it was quite close to a straight line. Consequently, failures during the testing phase occur following an inhomogeneous Poisson process, whose intensity can be derived as follows. First, due to the aforementioned observation, the cumulative number of failures, N(t), can be written as

$$\log N(t) \approx \log \alpha + \beta * \log t, \quad (1)$$

being α and β the parameters of the model. Hence, the cumulative failure intensity, Z(t), is modelled as

$$Z(t) = \alpha * t^{\beta}, \quad (2)$$

and the failure intensity, z(t), is easily derived as

$$z(t) = d/dt(Z(t)) = \alpha * \beta * t^{(\beta-1)}. \quad (3)$$

Subsequently, the estimation of α and β (and thus the software operational failure intensity at the end of the debugging process) is straightforward if the number of software failures and the testing time are known. Basically, this estimation can be done by direct fitting on the log-log plot, or by means of maximum likelihood estimation.

B. *OLT Software Dependability*

As mentioned before, both the number of software failures as well as the testing time is needed in order to predict the software operational failure intensity. Notably, the software dependability analysis in this paper builds on the results presented in [9].

Basically, the authors in [9] report the results of applying a regression testing technique to a GPON OLT software during more than one year. Additionally, not only the number of software failures is reported, but also the distribution of these failures over the testing time. Hence, when applying Duane's model to these results, it is possible to obtain a reasonable estimation of the GPON OLT software failure intensity as a function of the testing time.

After analyzing these results, the cumulative number of failures versus the cumulative testing time log-log plot is presented in Fig. 2 with red dots. In the figure, the cumulative testing time has been normalized in hours. Additionally, Fig. 2 also shows the fitted cumulative failure intensity (Z(t)) according to Duane's model, in blue. Particularly, the values of α and β for the fitted cumulative failure intensity are 0.311543 and 0.761157 respectively. Consequently, by substituting these values in (2), the fitted cumulative failure intensity of OLT software failures follows

$$Z(t) = 0.311543 * t^{0.761157}, \quad (4)$$

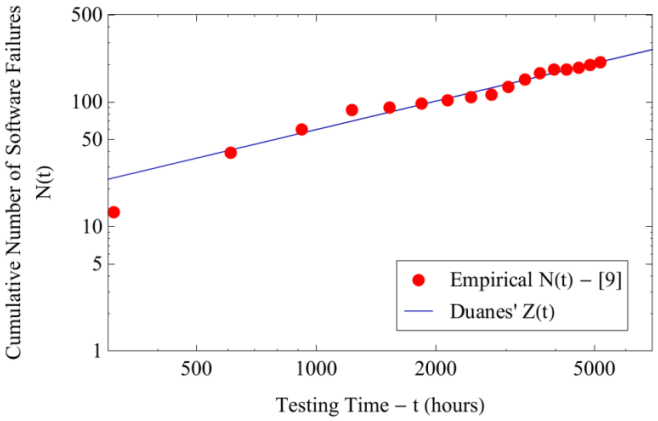

Fig. 2. Cumulative number of software failures as a function of the testing time (Data from [9]) and the Dunae's Z(t) fitted estimation.

while the software failure intensity follows

$$z(t) = d/dt(Z(t)) = 0.237133 * t^{-0.238843}. \quad (5)$$

By employing (5), the software failure intensity can be calculated for different values of the cumulative testing time. Finally, this failure intensity is used in the next section to introduce software failures into the Markov cost model assessing the dependability and failure costs of PONs.

IV. Markov Cost Model for PON Dependability and Dependability-Related OPEX

In this section, the Markov cost model employed to analyze both the dependability and dependability-related OPEX of PONs is introduced. First, the Markov cost model for hardware failures is presented. Then, this Markov model is modified with the software failure intensity calculated in Sect. III to include software failures in the dependability-cost analysis.

Briefly, Markov cost models allow for including cost considerations into Markov models, so that both dependability and failure-related costs can be calculated at the same time [11]. Markov cost models stem from the notion of Markov reward models, which associate a reward rate c_i with each state *i* of the Markov model. When dealing with failure costs calculations, the rewards are the cost rates (cost per unit time) related to failures in the corresponding state. As in [13], two dependability-related costs are considered in this study: costs related to failure repair and costs of paying penalties. Decidedly, the cost rate of a given state consists of two terms: the Repair Cost Rate (RCR) and the Penalty Cost Rate (PCR).

Regarding the RCR, failure repair costs are directly related to the repair crew's salary, the number of operative repair crews and the repair time. Subsequently, the RCR in a given state *i* is proportional to the crew's salary (S – in $/hour) and the number of operative crews in state *i* (OC_i):

$$RCR_i = S * OC_i. \quad (6)$$

Concerning the PCR, penalty costs depend on the agreed penalty rate (PR), the number of failed clients (FC) and the disconnection time. Also, in order to account for the impact of reputation loss in case of large outages, an exponential impact factor, χ, is introduced. Thus, the PCR in a given state *i* follows

$$PCR_i = FC_i^{\chi} * PR. \quad (7)$$

Purposely, the impact factor χ allows for a smooth insertion of the loss of reputation due to failures into the PCR. It was first introduced in [13], in a similar way as in [5]. Mainly, the basic idea behind the impact factor χ is to increase the PCR if the number of clients affected by a failure is high, due to the impact of negative publicity. From an operator's point of view and considering the same time period, failures affecting a large number of clients at the same time have a bigger impact than a large number of independent, non-overlapping in time failures affecting a small number of clients. Succinctly, over a month, a single failure (e.g. a digging) affecting 5 000 clients is much worse than 5 000 failures affecting one client, occurring at different non-overlapping times over the same month. The former type of failure will lead to negative press releases. Intentionally, a value of 1.1 is proposed for χ. Then, if the number of failed clients is small, the PCR will be almost the same as if χ is not included. In fact, it is the same if there is only 1 failed client. On the other hand, when the number of failed clients is considerable, the PCR will grow larger. For example, the PCR of a failure affecting 10 000 clients is 2.5 times larger than the same failure without any impact factor.

A. PON Hardware Failures

Let us now consider the Markov cost model when only hardware failures are present. As there are no software failures in this case, it will be used as baseline to measure the impact of software failures on the dependability and failure costs.

In essence, the modelled system is the PON architecture depicted in Fig. 1. For illustration, a significant part of the Markov model is shown in Fig. 3. Namely, state definition depends on the type of failed element. As explained in Sect. II, the different elements of the PON architecture are OLT chassis, OLT ports, feeder fiber, RN chassis, splitters, distribution fiber and ONUs. Since the split ratio is fixed to 1:32, the number of ONUs is also 32 (denoted N in Fig. 3). Additionally, splitters in the RN are assumed to fail if the RN chassis fails, and the same applies to the OLT ports hosted in an OLT chassis. Failure rates for the different components are taken from [3] and [14]. Typically, the longer the fiber, the more likely it is to fail, thus the fiber failure rate in [3] depends on the length of the fiber. In this study, different values for the lengths of feeder and distribution fibers are considered, in order to model dense or sparse PON deployment scenarios, following the values of the studies in [6], [13]. Dense scenarios correspond to densely populated urban areas, where users are located close to each other. Thus, the length of the feeder fiber is fixed to 3.75 Km., while the length of the distribution fiber is fixed to 0.375 Km. Sparse scenarios correspond to suburban or rural areas, being the lengths of the feeder and distribution fibers 18.2 and 1.8 Km. respectively.

As for the number of failed clients in each state, it decidedly depends on the type of failed element. Following the values presented in [6], [13]; the number of clients affected by each type of failure has been fixed as follows. OLT chassis failures typically affect 1 600 clients, while RN chassis affect 100 clients. OLT ports and splitters only affect the 32 clients associated to the PON. Unequivocally, ONU failures affect only 1 client. Regarding fiber failures, the number of affected

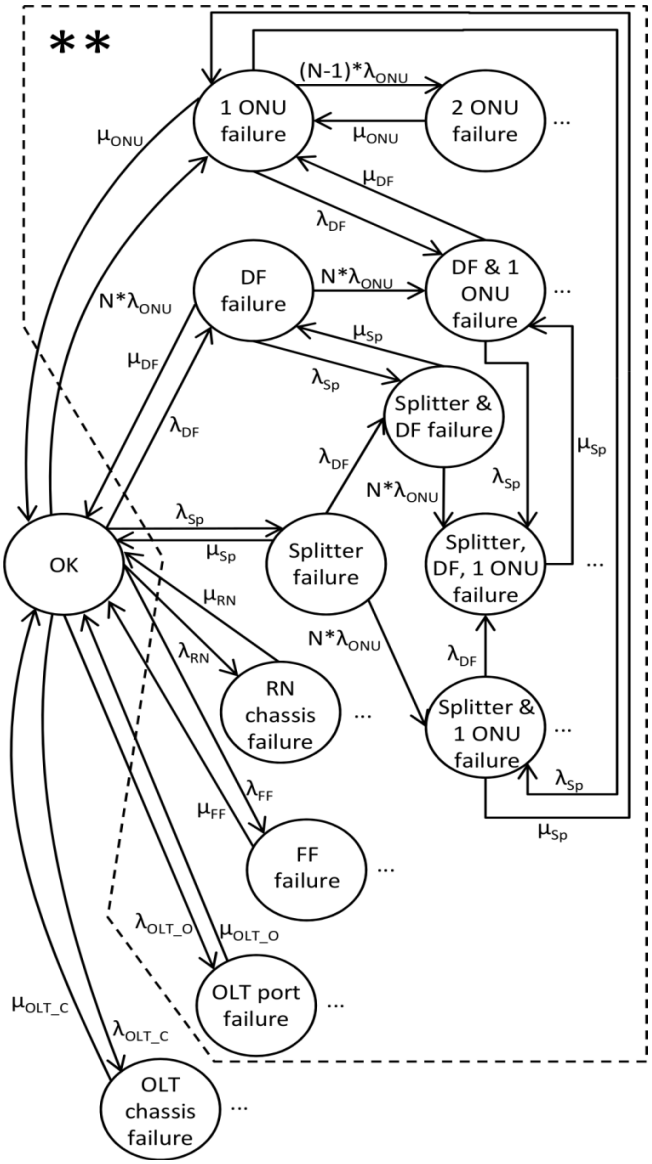

Fig. 3. Markov model considering only hardware failures.

clients is slightly more complicated to calculate. In general, fiber failures are related to diggings in the trenches containing the fibers. Hence, if the digging occurs very close to the CO, it will affect a large number of fibers, and thus a large number of clients. Contrarily, the number of affected clients will be smaller if the digging occurs far from the CO. To model this effect, the number of affected clients in case of feeder fiber failure is assumed to be a random variable uniformly distributed between 1 000 and 5 000 clients. In case of distribution fiber failure, the same reasoning applies, but the uniform variable is defined between 1 and 100 failed clients.

Finally, regarding the repair process, repair rates for the different elements are also taken from [3] and [14]. Besides, it is assumed that there is only one repair crew, as it was shown in [13] that one repair crew is enough to handle repairs in most situations. If there are two or more failed elements, the component leading to a higher reduction in the cost rate in a shorter repair time is repaired first.

B. OLT Software Failures Modelling

In order to include software failures into the model, two new states are introduced into the Markov model of Sect. IV A. Additionally to the data presented in Sect. III, the GPON OLT software failures reported in [9] are categorized into four different grades with respect to their criticality. Namely, these four categories are “low criticality bugs”, “medium criticality bugs”, “high criticality bugs” and “very high criticality bugs”.

Low and medium criticality bugs are stated to somewhat hamper the PON performance, but they do not lead to total crash or total service interruption. High and very high criticality bugs, on the other hand, interrupt system operation and affect basic functionalities. To model the latter, the state named “software failure” is introduced in the Markov chain, and it is assumed to be a down state as service is interrupted. Low and medium criticality bugs are modelled with the state called “excited software”. As there is no clear indication in [9] of how big the performance reduction becomes when these bugs occur, two different approaches are assumed. In the optimistic approach, the system is considered to be working when in the excited software state. Mainly, this means that although the performance may be reduced, end users do not notice it. Contrarily, the pessimistic approach assumes the excited software state to be a failed state, where part of the OLT software is down and end users do notice the outage.

A reduced version of the Markov model accounting for both hardware and software failures is depicted in Fig. 4. The dashed oval in Fig. 4 represents all hardware and hardware-software possible failure combinations, according to the states in the dashed box of Fig. 3. Please note that the state corresponding to OLT chassis failure is repeated in both figures for clarity.

In Fig. 4, λ_{soft} is the software failure intensity in (5), which depends on the testing time. When computing this intensity, all types of bugs were considered. However, high and very high criticality bugs are reported to account for a 33% of the total number of bugs. Thus, the intensity leading to the software failure state is multiplied by 0.33 (denoted p in Fig. 4). Decidedly, low and medium criticality bugs represent a 66% of

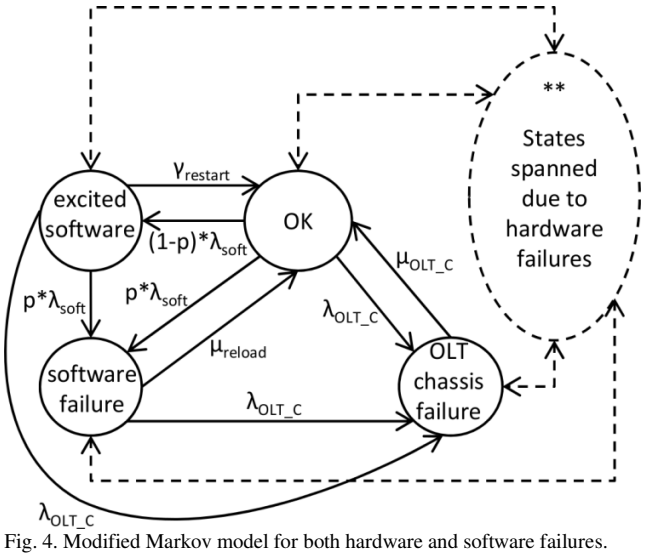

Fig. 4. Modified Markov model for both hardware and software failures.

the total software failures. Thus, the intensity leading to the excited software state is multiplied by 0.66 (1-p in Fig. 4). Noticeably, OLT software is assumed to be running on the OLT chassis. Hence, there cannot be software failures if there is an OLT chassis hardware failure (there are no transitions from the OLT chassis failure state to the excited software state or the software failure state). Besides, hardware repair of the OLT chassis assumes to fix also software failures, as the OLT chassis is switched off and on and the software brought to a consistent initial state.

Regarding software repairs, two different restoration actions are considered. A system in the excited software state is assumed to be brought back to a free-failure state by a restart, while a system in the software failure state requires a full reload of the system. When a restart is initiated, the process with the failure is stopped, a subset of the processor's data reset, and the processing resumed. Because the bugs in the excited software state have not a high criticality, the restart is a quick process (5 minutes, $\gamma_{restart}$ = 1/12 h^{-1}), and negligible human intervention is assumed (there is no impact to the RCR).

On the other hand, a reload is a more complex action and requires more time, as the software failures in this case are more severe. During a reload, the processor and peripherals are reset and tested, while the processor's software and data are reloaded. The average duration of this action is assumed to be 30 minutes (μ_{reload} = 1/2 h^{-1}), and human intervention is not negligible (it does affect the RCR). Note that software restart/reload can be done in parallel with hardware repairs – i.e. there is one dedicated repair crew for hardware and another for software failures.

Finally, the number of affected clients due to software failures is fixed as follows. In the software failure state, the OLT chassis system is considered to be down, thus this kind of failure affect 1 600 clients. When the system is in the excited software state, the number of failed clients depends on the chosen approach. Because the software is assumed to be working in the optimistic approach, there are no failed clients in this case. Yet, the pessimistic approach considers that some end users do notice the outage, thus the number of failed clients is modelled as a uniform variable between 1 and 400.

V. Dependability and Failure-Related OPEX Simulation Results

This section presents the results of the dependability and failure-related OPEX study, after solving the Markov models presented in Sect. IV by means of simulations. After a brief description of the simulator, results for the availability, failure impact and dependability-related OPEX are reported.

Due to the large number of states, simulations are required to solve the Markov models. Amid other options, a uniformized simulator [15] has been implemented; solved by means of Monte Carlo simulation because of its flexibility and easy implementation [11]. As explained in Sect. IV. A, two different scenarios have been considered, namely dense and sparse scenarios. Additionally, two different approaches for the excited software state are assumed as explained in Sect. IV B – the optimistic and the pessimistic approach. The effect of the testing phase duration (which affects λ_{soft} as described in Sect. III) is also investigated by varying the testing time from 5 500 (duration of testing phase in [9]) to 50 000 hours. Finally, results also include the case with only hardware failures as baseline to measure the hampering of software failures in the PON dependability and failure-related OPEX.

A. *Asymptotic Availability*

Regarding asymptotic availability, Fig. 5 shows the results for dense scenarios, while Fig. 6 depicts the results for sparse scenarios. Results are presented with 95% confidence intervals, although most of them are hidden behind the marked points. Undoubtedly, results when software failures are not present do not depend on the testing time.

Let us focus first on dense scenarios (Fig. 5). Decidedly, results show that software failures markedly dominate the PON system availability, both for the optimistic and pessimistic approach. The effect of high and very high criticality bugs is the difference between the no software and the optimistic approach curves – notably dominating the availability reduction. Besides, the effect of the low and medium criticality bugs (if assumed to cause a failure) is the difference between the optimistic and pessimistic curves. Even if the testing phase is notably large (50 000 hours), availability drops from 0.99975 (no software) to 0.9968 (optimistic) or 0.9957 (pessimistic).

As the software testing time is increased, the availability also increases for both approaches, as the software failure intensity decreases following Duane's model. Yet, this increase is more remarkable with testing times between 5 500 and 20 000 hours. Asymptotically, both approaches tend to the no software case, as software failures become negligible.

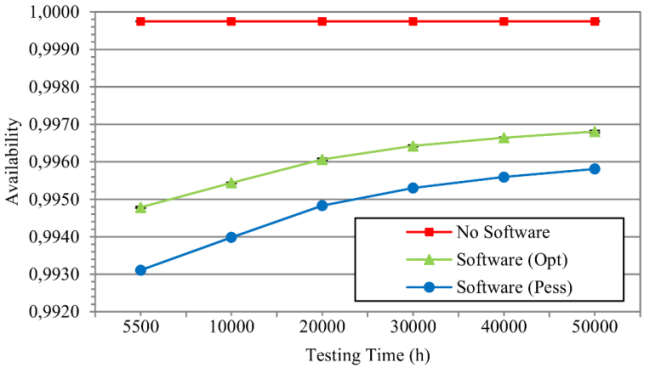

Fig. 5. Availability results for dense scenarios.

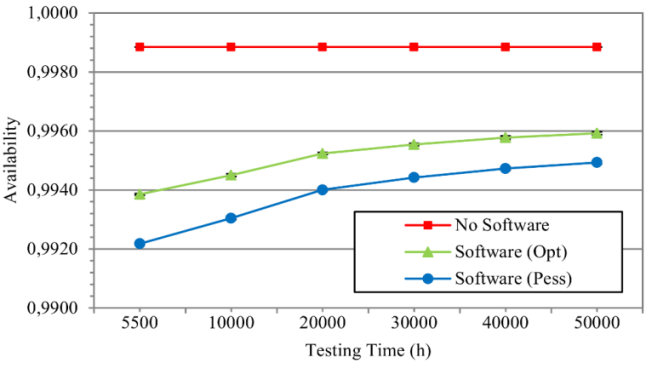

Fig. 6. Availability results for sparse scenarios.

As for the availability in sparse scenarios, depicted in Fig. 6, roughly the same considerations are shown. Software failures also dominate the availability, although in a lesser way than in dense scenarios, with the biggest reduction due to high and very high criticality bugs. Besides, the biggest reduction in the effect of software failures is achieved during the first 20 000 hours as in dense scenarios. Yet, the availability is also noticeably affected by hardware failures due to larger feeder and distribution fibers. In the best case (50 000 testing hours), the availability is reduced from 0.99884 (no software) to 0.9959 (optimistic) or 0.9948 (pessimistic), where approximately a reduction of 0.00091 with respect to the dense scenario correspond to hardware failures (fiber infrastructure).

B. Failure Impact

As for the failure impact results, Fig. 7 and Fig. 8 show the Cumulative Distribution Function (CDF) of the cumulative number of failed clients for dense and sparse scenarios. Plainly, these figures show the probability of a failure causing less than or equal to a given number of failed clients (i.e. failure impact). Figures depict not only the case with only hardware failures, but also the optimistic and pessimistic approaches with the smallest and highest testing times. Curves for other testing times lie in between these two and are not presented for clarity.

In dense scenarios, Fig. 7, it is clear that concerning hardware failures, feeder fiber failures dominate the failure impact. The cumulative probability of failures with small impact is modest till 1 000 failed clients, when it starts increasing due to the effect of feeder fiber failures. When software failures are included, all cases present an increase in the cumulative probability around 1 600 failed clients. Notably, this probability corresponds to the software failure state, dominating the failure impact. Markedly, the probability decreases as the testing time increases (solid versus dashed curves) as the software becomes less failure-prone. Finally, the difference between the optimistic and pessimistic approach is also notable. In the optimistic approach (green curves), the probability of failures with small impact is scant as the excited software state does not contribute to the failure impact. Yet, this probability is noteworthy in the pessimistic approach; although still the high and very high criticality bugs dominate.

Results in sparse scenarios, Fig. 8, present almost the same concerns as dense scenarios. However, fiber failures become more notable in this case. Especially, the probability of failures with more than 1 000 failed clients increases considerably, due

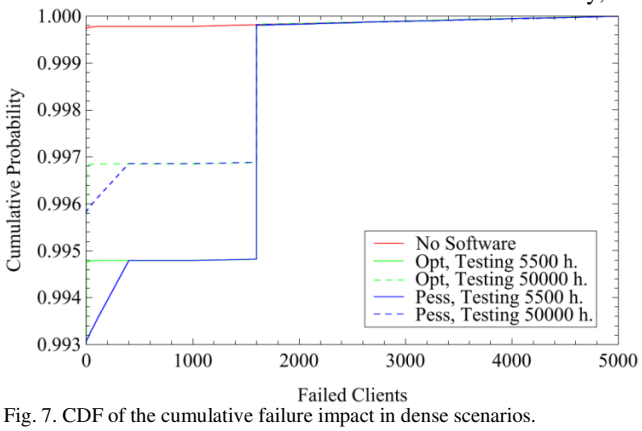

Fig. 7. CDF of the cumulative failure impact in dense scenarios.

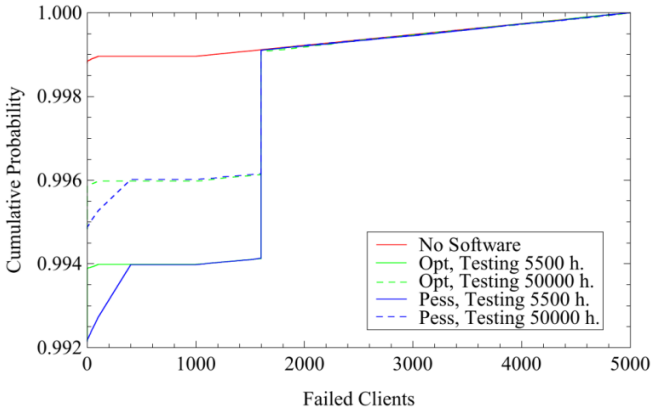

Fig. 8. CDF of the cumulative failure impact in sparse scenarios.

to larger feeder fibers. In fact this probability is now comparable with that of the excited software state in the pessimistic approach, but with a much larger failure impact. Besides, the probability of hardware failures with small impact is also noticeable, caused by distribution fiber failures.

C. Dependability-related OPEX

Finally, dependability-related OPEX costs are presented as expected cost per client (in $) over a time span of 1 year. The expected cost is assessed by multiplying the Expected Cost Rate (ECR) by the time span of interest. The ECR comes from the cost rate of each state (c_i) and their probabilities (p_i) as

$$ECR = \sum_i c_i * p_i \quad (8)$$

Besides, expected costs are broke down into Expected Repair Costs (ERC) and Expected Penalty Costs (EPC) as the cost rate of each state consists of the RCR and the PCR. As parameters for the cost analysis, the repair crew's salary is fixed to 190 $/hour and the penalty rate to 10 $/hour. Results are presented with 95% confidence intervals.

Fig. 9 shows the expected costs per client in dense scenarios. Decidedly, repair costs are almost negligible with respect to penalty costs. As expected from previous results, software failures remarkably increase the expected costs. Among software failures, high and very high criticality bugs (optimistic approach) produce the biggest increase, due to their large failure impact (1 600 clients) and larger repair process. The expected cost increase due to low and medium criticality bugs (pessimistic approach) is almost comparable to the cost of hardware failures. Still in terms of costs, the latter is bigger due

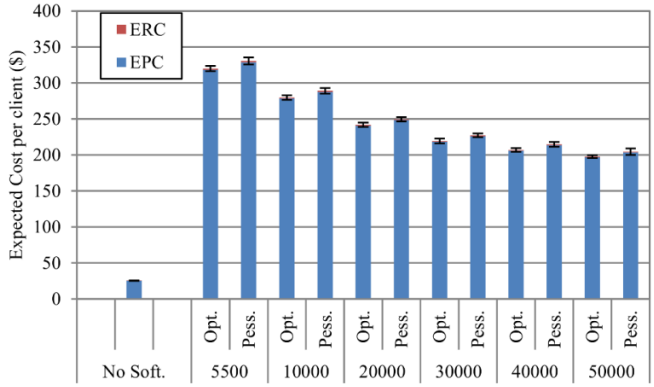

Fig. 9. Expected cost per client in $ for dense scenarios in 1 year time span.

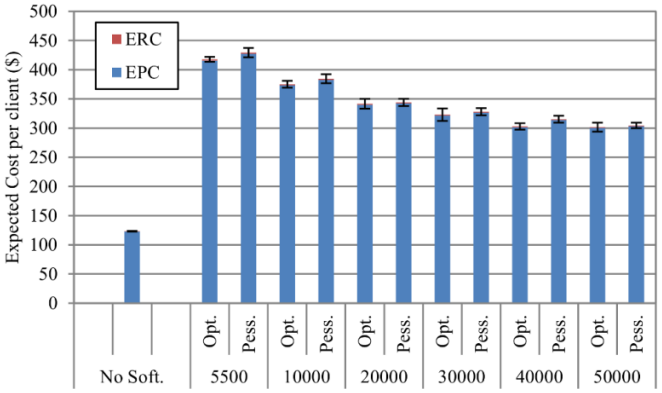

Fig. 10. Expected cost per client in $ for sparse scenarios in 1 year time span.

to the large failure impact of feeder fiber failures. When the testing time is increased, the costs are also reduced. Yet, this reduction grows smaller for testing times larger than 20 000 hours, as hinted from the availability and failure impact results.

In sparse scenarios, Fig. 10, the same trends can be identified. As larger fibers become more failure-prone, there is an increase in the expected cost because of fiber infrastructure failures. Especially feeder fiber failures, because of their large failure impact and long repair time, lead to a large increase in the expected costs. Contrarily, the expected cost due to low and medium criticality bugs is now of minor importance, as the expected costs of the optimistic and pessimistic approach are almost the same. High and very high criticality bugs are still the most significant, being the biggest contribution to the expected costs. As before, increasing the testing time beyond 20 000 hours does reduce the costs, but in a marginal way.

VI. Conclusions

In this paper, a thorough dependability and failure-related OPEX study of TDM PONs has been performed. While software failures have been the main object of the study; hardware failures have also been included for the sake of completeness. By applying Duane's model for reliability growth to the results in [9], the dependability (more precisely, the failure intensity) of the OLT software as a function of the testing time has been estimated. Subsequently, a Markov cost model accounting for both hardware and software failures has been developed. Finally, the asymptotic availability, failure impact and dependability-related OPEX of these two types of failures in TDM PONs have been assessed. While high and very high criticality bugs are considered to cause a failure; two different approaches have been considered regarding low and medium criticality bugs, namely optimistic and pessimistic approach. The optimistic approach assumes that low and medium criticality bugs do not cause a failure, while these bugs do cause a failure in the pessimistic approach.

Primarily, this paper shows that software failures present a significant threat for PONs dependability and OPEX in both dense and sparse scenarios. High and very high criticality bugs not only seriously hinder the availability, but their large failure impact considerably increases the failure-related OPEX. In dense scenarios, low and medium criticality bugs, when considered to cause a failure, hamper the availability in a greater way than hardware failures. Yet, their relatively small failure impact and quick repair make their effect with respect to failure-related costs comparable to hardware failures in dense scenarios. Contrarily, in sparse scenarios, low and medium criticality bugs pose the same threat to availability as hardware failures. Yet with respect to failure-related OPEX in sparse scenarios, hardware failures are far more serious than low and medium criticality bugs. Mainly, this occurs because hardware failures are typically related to fiber cuts, with a large failure impact and repair time. Expectedly, the effect of software failures can be reduced by increasing the duration of the software testing time. However, this reduction becomes marginal for testing times greater than 20 000 hours.

Finally, the results presented in this paper call for further research. Due to the lack of information regarding software failures in PONs, software failure classification and description as well as the software dependability modelling is minimal. Also, a proper cost analysis of software failures requires modelling of the resources and costs associated to the testing phase. This way, a precise analysis of the trade-off between testing phase duration and costs versus the effect of software failures in service provision can be performed.

References

[1] F. Effenberger *et* al., "An introduction to PON technologies," *IEEE Commun. Mag.,* Vol. 45, Issue 3, pp. S15 – S25, Mar. 2007.

[2] G. Kramer, M. De Andrade, R. Roy and P. Chowdhury, "Evolution of optical access networks: architectures and capacity upgrades," *Proc. of the IEEE,* Vol. 100, pp. 1188 – 1196, May 2012.

[3] J. Chen, L. Wosinska, C. M. Machuca and M. Jaeger, "Cost vs. reliability performance study of fiber access network architectures," *IEEE Commun. Mag.*, Vol. 48, Issue 2, pp. 56 – 65, Feb. 2010.

[4] E. Wong, "Survivable architectures for time and wavelength division multiplexed passive optical networks," *Journal of Optics Communications*, Vol. 325, pp. 152 – 159, Apr. 2014.

[5] A. Dixit *et all*., "Protection strategies for next generation passive optical networks -2," *Proc. of ONDM 2014*, Stockholm, Sweden, May 2014.

[6] A. Fernandez and N. Stol, "Protecting PONs: a failure impact, availability, and cost perspective based on a geometric model," *Proc. of DRCN 2014*, pp. 1 – 8, Ghent, Belgium, Apr. 2014.

[7] D. Oppenheimer and D. A. Patterson, "Architecture and dependability of large-scale internet services," *IEEE Internet Comput.*, Vol. 6, Issue 5, pp. 41 – 49, Sept./Oct. 2002.

[8] L. A. Barroso, J. Clidaras and U. Hölzle, "The datacenter as a computer: an introduction to the design of warehouse-scale machines," 2nd edition, *Synthesis series on Computer Architecture*, Morgan & Claypool Publishers, pp. 101 – 113, May 2013.

[9] A. C. Fadel, R. Moraes and E. Martins, "Automated validation of embedded optical network software," *Proc. of LADC 2011*, Sao Jose dos Campos, Brazil, Apr. 2011.

[10] M. R Lyu (ed.), "Handbook of Software Reliability Engineering," Ch. 3, pp. 98 – 99, McGraw-Hill/IEEE Comp. Soc. Press, Apr. 1996.

[11] G. J. Anders and A. M. Leite da Silva, "Cost related reliability measures for power system equipment," *IEEE Trans. Power Syst.*, Vol. 15, No. 2, pp. 654 – 660, May 2000.

[12] ITU-T Rec. G984.1, "Gigabit-capable passive optical networks (GPON): general characteristics," Mar. 2008.

[13] A. Fernandez and N. Stol, "OPEX simulation study of PONs based on a network geometric and Markov cost models," *Proc. of ONDM 2014*, Stockholm, Sweden, May 2014.

[14] OASE Project, "D4.2.1: Technical Assessment and comparison of next-generation optical access systems concepts," Oct. 2011.

[15] S. M. Ross, "Introduction to probability models," 10th edition, Ch. 6, pp. 406 – 408, Academic Press, Oxford, United Kingdom, 2010.