Validation of GPON Networks Using a Fault-Injection Based Approach

Aline Cristine Fadel[1], **Regina Moraes**[1], **Paulo Martins**[1], **Eliane Martins** [2]

[1]Faculdade de Tecnologia (FT)
University of Campinas (Unicamp) – Limeira, SP – Brazil

[2]Instituto de Computação (IC)
University of Campinas (Unicamp) – Campinas, SP – Brazil

{aline,regina,pmartins}@ft.unicamp.br, eliane@ic.unicamp.br

Abstract. *This paper presents a software validation approach for a Gigabit Passive Optical Network (GPON) system. We applied a fault injection technique to GPON software, more specifically to both the Optical Line Termination (OLT) software and the communication protocol (OMCI). Two campaigns were performed: The first consists of automated fault injection validation based on state machine model. The second uses fault injection at the communication protocol level. More specifically, it inserts incorrect protocol data unit (PDU) fields to transmitted messages. This latter campaign integrates the CoFI methodology that is guided by standards focused on test cases execution and protocol implementation. Those standards provide rules for the correct operation of the system and are used to plan the test cases. The results emphasize the relevance of the approach: 1) The effectiveness of the tests applied is evident when compared with a one year and half campaign previously used; 2) The test cases using the proposed approach reveal significant number of failures in a short period of time, and 3) Despite of the long term testing using traditional methods, high criticality faults were only disclosed by applying the proposed approach.*

1. Introduction

GPON networks have gained ground in the telecommunications market, increasing users' expectations towards the quality of services provided. In order to fulfill these expectations, these networks must provide a high degree of dependability, as their availability must reach at least 99.999 %. Such rate corresponds to a downtime of up to 5 minutes per year [RAMASWANI and SIVARAJAN 2002]. In addition, these networks must meet low transmission error and high transfer rates requirements. Clearly, these requirements can only be met if appropriate techniques are in place to guarantee the specified level of dependability performance.

The interruption of communication services and data losses are common failures in telecommunication networks. They increase the risk of system unavailability and have a negative impact on both customers and providers. One solution to minimize the occurrence of failures lies in the implementation of fault tolerance mechanisms. The goal of fault tolerance is to maintain the operation of a system despite the presence of faults.

However, the bare installation of these mechanisms is not enough: the fault tolerance mechanisms must be checked prior to their implementation.

To experimentally assess the dependability of computer-based systems, the dependability community has a long tradition of testing computer systems based on fault injection [Arlat et al. 1990]. It allows the evaluation of the system behavior when faults are deliberately introduced into the system [HSUEH et al. 1997]. Fault injection also facilitates the monitoring of experiments, due to the prior knowledge of where faults are placed. The combination of fault injection and finite state machine models was efficiently used on space applications [Ambrosio et al. 2006]. In one of the experiments reported in their work, the CoFI methodology (Conformance and Fault-Injection) was used. CoFI integrates two test approaches: conformance testing and fault injection. COFI methodology is used to generate test cases and fault cases based on finite state machines [Ambrosio et al. 2006].

The goal of this work is to report the results of an experimental study to complement tests techniques previously applied on GPON network (regression tests, white box tests). In addition, fault injection technique was applied to validate the GPON embedded software, ensuring the dependability level required from these telecommunication systems. In particular, this work is based on a GPON developed by CpQD [CPqD 2013]. GPON networks are standardized by ITU-T [ITU-T-G.984.1 2008] recommendations and they provide the infrastructure for triple-play (voice, data and TV) service.

This work offers contributions for practitioners developing embedded software systems that need to meet more stringent availability and reliability service requirements. This study was able to demonstrate that the testing techniques applied were effective for validating the system in an adverse environment. The contribution of this work is an approach that is based on the following concepts:

- *Fault injection*: it emulates physical faults and automate the execution of tests in the presence of these faults, i.e. to apply systematic fault injection at the state transition level of a GPON software;
- *Interface testing:* it tests the robustness of the OLT (Optical Line Termination) software when OMCI packets (ONT Management and Control Interface) are corrupted;
- *State-machine transition testing*: it tests the GPON at the state transition level. It allows for optimal testing coverage and repeatability;
- *Test automation*: it is used to both emulate and inject faults in the GPON system. A test robot was designed and implemented to execute regression testing and later adapted to perform fault injection.

We argue that this approach is a suitable one since it tackles and exercises two key aspects of GPON systems: 1) their dynamic behavior is exercised by means of the automated fault-injection at the state transition level, and 2) their service (functionality and performance) provided is also addressed by means of fault-injection at the component interface level.

The remainder of this paper is organized as follows: Section 2 introduces the validation model and describes two testing campaigns; Section 3 presents the results obtained; Section 4 discusses the results and our conclusions are presented in Section 5.

2. Validation Model

In this work, fault injection is exploited in two test campaigns (Fig. 1): 1) In the first campaign, physical faults are emulated using the SoftWare Implemented Fault Injection technique (SWIFI) and systematically injected at all of the transitions of the device's state machine; 2) In the second campaign, the faults are injected manually in the communication protocol by corrupting the OMCI protocol fields to emulate a malformed message. In both cases, the tests are applied to the GPON system represented in Fig. 2 along with a test robot (see Section 2.1). It consists of the following major components:

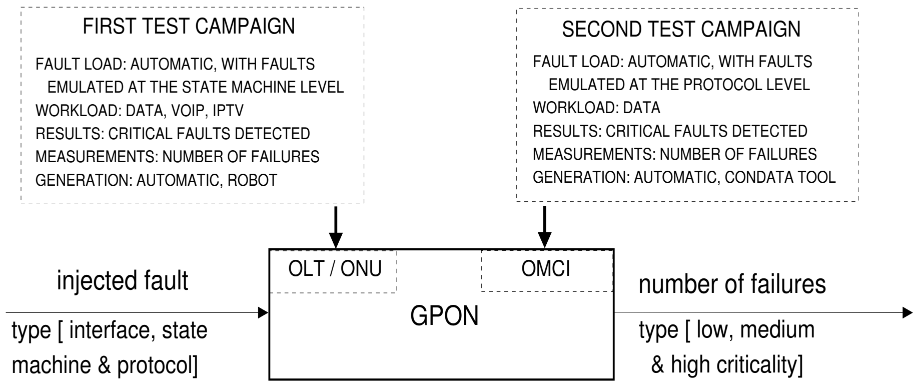
FIRST TEST CAMPAIGN
FAULT LOAD: AUTOMATIC, WITH FAULTS EMULATED AT THE STATE MACHINE LEVEL
WORKLOAD: DATA, VOIP, IPTV
RESULTS: CRITICAL FAULTS DETECTED
MEASUREMENTS: NUMBER OF FAILURES
GENERATION: AUTOMATIC, ROBOT
SECOND TEST CAMPAIGN
FAULT LOAD: AUTOMATIC, WITH FAULTS EMULATED AT THE PROTOCOL LEVEL
WORKLOAD: DATA
RESULTS: CRITICAL FAULTS DETECTED
MEASUREMENTS: NUMBER OF FAILURES
GENERATION: AUTOMATIC, CONDATA TOOL
injected fault
type [interface, state machine & protocol]
OLT / ONU
OMCI
GPON
number of failures
type [low, medium & high criticality]

Figure 1. Validation model.

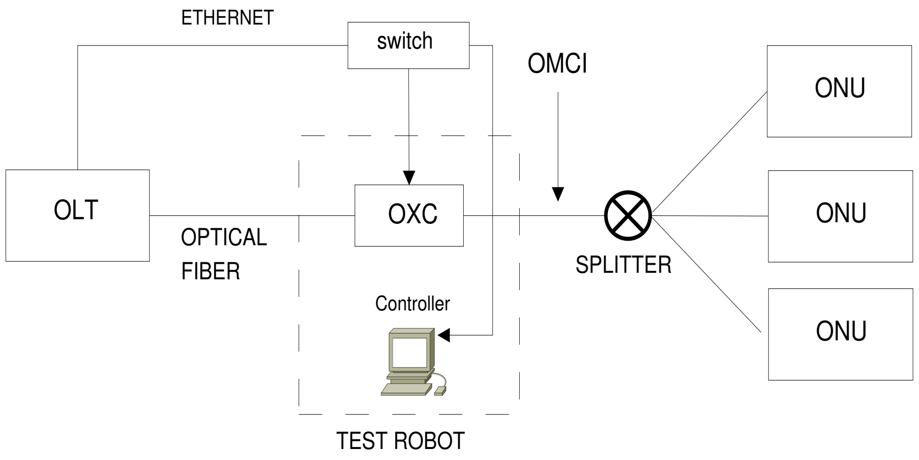

Figure 2. The GPON System and Test Robot - Physical View

- *Optical Network Unit (ONU):* This is the equipment placed at the user premises (i.e. houses, building);
- *Optical Line Termination (OLT):* This is the equipment placed at the service provider premises. The OLT has a communication interface with OMCI protocol ($COM1$), which in turn communicates through another interface ($COM2$) connecting to ONUs. The OLT has another communication interface, the *Command Line Interface (CLI)*, used by the network operators to interact with the GPON system;
- *ONT Management and Control Interface (OMCI)*: This is the communication protocol used in GPON networks. The OMCI controls the communication among the

OLT and the ONUs and vice versa. This protocol is specifically standardized at G.984.4 [ITU-T-G.984.4 2008]. The OMCI protocol consists of a set of messages including *create, delete, set, get*, and MIB *upload*. These messages are created by the OLT.

Faults are synthetically generated and introduced at different points of the system, i.e. interface or state transitions. Fault injection was the technique of choice due to the fact that this approach provides relevant support to reproduce failures that may occur during the operational phase. The goal of both test campaigns is to observe the number of failures that result after the faults are injected. A large number of failures indicates a poorly designed system and/or a successful validation approach. A small number of failures may indicate a system that is either adequately tested or a validation method that is inefficient.

Prior to this work, these tests were manually performed by simply removing and reinserting the fiber that connected the ONU to the system. However, this approach had two major drawbacks: 1) The repeatability of the experiment was compromised, as the exact time of fiber's removal/reinsertion was unknown, and 2) It was a challenge to measure the coverage of these tests.

2.1. First Test Campaign - Fault Injection in the State Machine (OLT and ONU)

The first test campaign uses fault injection to emulate interface faults in order to check the behavior of the OLT and ONU software when a communication failure occurs and/or the GPON devices are shut down. Specifically, by monitoring the behavior we also validate the fault-tolerance mechanisms implemented by the OLT software. The failures emulated by injected faults model real faulty scenarios that may occur in a system. For example, the fiber can be interrupted, the hardware of the ONU or OLT can be damaged or there may be power outaged. In these situations, the impact on the users is inevitable, and these problems must be addressed as soon as possible so that the impact of the failures is minimized. In an ideal system, after the occurrence of such failures, the network is either subject to maintenance (the fibers or the devices can be replaced) or the signal is recovered and the connection among the devices is restored. In this latter case, the OLT and ONUs must return to their previous state, i.e. before the interruption of communication, and it must re-establish communication without any human intervention. The solution found was the automatic emulation of communication loss using fault injection, forcing the faults to be injected at the transitions of the state model. In this campaign, the tests cases were manually generated, but automatically injected. The state-machine model represents the allowable behavior of the system, and test scenarios were derived from each transition between states. 160 test cases were generated based on 14 state machines. In these test cases, a fault was injected (e.g. the fiber interruption) at every state transition, until the whole state model was covered.

The first workload (i.e. the functional activity) for this campaign is composed solely by data transmission. A test robot was adapted to control the execution of test cases and to automatically control the injection of faults (Fig. 2). It contained an optical switch (OXC - Optical Cross Connection) which was responsible for the interruption of communication among the devices , allowing the fault to be injected. An Ethernet switch was used to connect the robot controller, the OXC and the OLT.

After the execution of the tests cases, the robot generates a report with all results (i.e. number of faults and their criticality), separating the successful test cases from the failed ones. The Winlogger, a proprietary tool developed by CPqD, is reponsible for monitoring the experiments, and it also receives and classifies the reports (logs). The network topology is changed during the tests execution (i.e. the number of ONUs connected and the number of services available). However, the results obtained when the topology changed are all very similar, i.e. the failure has occurred independently of the workload. The period that the system was interrupted by the optical switch had no impact either on the results (for example, they were the same for a two-second and a two-minute interruption). These identical results were obtained because the state machines were stuck in the same state waiting for new input.

In order to further complement the tests described so far, we introduced new experiments with voice (VoIP) and TV (IPTV). The same state machines were used, despite the different type of traffic being used. This is due to the fact that GPON presents a similar behavior regardless of the type of traffic being transferred. Therefore, no new transitions were needed.

The behavior of both VoIP and IPTV was also analyzed when the ONU is subject to communication faults. To emulate these faults, the fiber connecting the target ONU was removed. To test the system with VoIP data, the fiber was removed when a VoIP call was running. When the fiber was reinserted, the ONU was back to its state before the crash and the phone connected to the ONU presented a new dial tone, but the previous connection was lost. To test IPTV data, the fiber was disconnected when a TV channel was broadcasting data. The video stream was interrupted until the ONU returned to its prior state, i.e., the state prior to the fiber disconnection. The video restarted to be transmitted, but the content that was being transmitted while the ONU was disconnected was lost. These results are the expected ones in a triple-play network.

2.2. Second Test Campaign - Fault Injection in the Message Fields (OMCI)

The goal of this second campaign is to evaluate whether the OLT software is fault-tolerant when the messages received from the OMCI protocol are corrupted. Therefore, the faults were manually injected before transmission in the following OMCI packet fields [ITU-T-G.984.4 2008]: *MsgType, ClassID, EntityID and CRC*. In this case, the injected software faults emulate communication faults by introducing malformed messages.

The CoFI methodology [Ambrosio et al. 2006] was used in this campaign to guide the GPON testing due to the clear definition of its steps. CoFI has been successfully applied to space applications. However, in this work the methodology was applied to a new context, i.e. embedded software for telecommunications. It proved to be robust due to the creation of state machines and the generation of test cases and faults, including fault injection. The methodology also allows planning of the testing activity from the beginning of the system development process, adding more quality to the final product.

The CoFI methodology consists of 6 steps: The first step defines the scope of testing, i.e. in this case the OMCI protocol. The second step generates the state machines, which in this work are based on the OMCI ITU-T standard. In the third step the test cases are generated. This was the only automated step in this work, and it was carried out by the ConData tool [MARTINS et al. 1999]. The execution of the test campaign is defined by the fourth step. It consisted in sending and receiving messages to and from the ONU. One of the OMCI operations that was tested is the *MIB upload*. This process represents the synchronization of the ONU once it is activated. Another operation that was explored was the software *download*, which is used to update the ONU software. It also includes the activation and initialization of the new software version, which effectively executes the ONU software update. The fifth step is an analysis of the test results, and the sixth step deals with report generation.

The system under testing in this second campaign is the same (GPON system, presented in Fig. 2), as well as the monitoring tool, Winlogger, which is also used in this second campaign. The messages received by the OLT are dynamically intercepted and realistic internal software faults of this environment are injected in order to validate the OLT's robustness. This is accomplished by adding a new software module to the protocol machine implementation that receives, corrupts and then relays the OMCI messages to the OLT.

These realistic software faults (internal faults) are statically injected (i.e. offline injection) into the OMCI protocol and several faulty versions are generated. These versions are used in the experiments corrupting the messages received by the OLT, so that the OLT robustness is validated.

Sixty test cases were generated based on the state machines created. All the test cases have the OMCI message changed, due to errors inserted into any message that should be executed by OLT (fifth step of CoFI). This campaign was the first effort to test the OMCI interface when messages are sent to the OLT. Prior to the execution of this campaign, there was no mechanism that allowed this kind of robustness testing.

3. Results

The results obtained in the first campaign are shown in Table 1. The state machines are presented in the first column and the second column shows the number of transitions of each state machine. The third and forth columns present the number of failures observed, which are classified as medium (impact the system performance but do not interrupt the service) and high criticality (can interrupt partially the system or seriously impact the network performance) respectively. The last column shows the total number of failures observed regardless of their criticality. Most of the failures were classified as medium criticality. However, high criticality failures were also identified, which could have interrupted the system or compromised its normal performance during the operational phase.

It is worth noticing that the state machines identified as D and E presented more failures than any other. One explanation for this occurrence is the complex interaction that these machines have with other machines, and therefore they are considered more critical. In the state machine D, highly critical faults were found, causing the system to crash.

Another representation of the failures observed in this first campaign is presented in Fig. 3. It is possible to observe that 80 % of the test cases presented failures as a result.

The failures observed in the first campaign were undetected in all tests performed prior to this work, including regression testing (details can be found in [FADEL et al. 2011]). The failures found previously by using other techniques were less

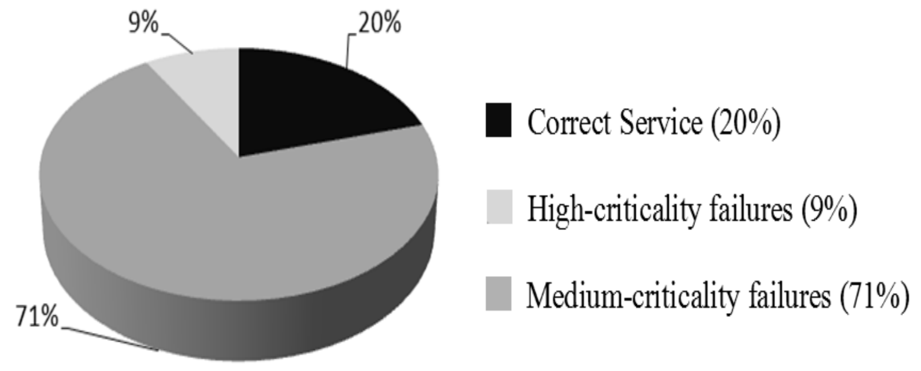

Figure 3. Results of the first campaign

critical and belonged to another class, such as failures caused by misinterpretation of the commands requirements. They were also easily identified once regression testing was applied.

Table 1. Distribution of faults (1^{st} campaign)

State Machine Id	Number of Transitions	Medium Criticality Fault	High Criticality Fault	Total Number of Faults
A	12	2	0	2
B	4	3	0	3
C	3	0	0	0
D	22	6	2	8
E	12	11	0	11
F	4	2	0	2
G	6	0	0	0
H	8	0	0	0
I	24	0	0	0
J	33	6	2	8
L	12	2	1	3
M	4	1	1	2
N	12	0	4	4
O	4	0	4	4

For the second campaign, 40 test cases were executed, i.e. 10 per type of message (Table 2). The first column shows the OMCI field where the faults were injected. The second column shows the amount of tests where the system delivered correct service (i.e.the injected fault is ignored). The next columns show the number of faults detected. They are classified by their criticality as medium, high and very high degree (the system crash). For example, for faults injected at the CRC field, the system exhibited correct service in nine out of ten cases, and only one fault was deemed to be a very-high criticality fault. Similarly, for faults injected at the $MsgType$ field, 7 faults were deemed to be of medium criticality, three were highly critical and no fault belonged to the very-high critical category.

Fig. 4 presents the results obtained in the second campaign. It is possible to observe that only 22 % of the tests presented successful results; failures were present in

Table 2. Distribution of the number of faults detected (2^{nd} campaign)

Tests	Correct Service	Incorrect Service		
		Medium-Criticality	High-Criticality	Very-High Criticality
MsgType	0	7	3	0
ClassID	0	4	2	4
EntityID	0	5	1	4
CRC	9	0	0	1
Total	9	16	6	9

the majority of them. These results show that OLT was not tolerant to faults introduced in the OMCI message packets.

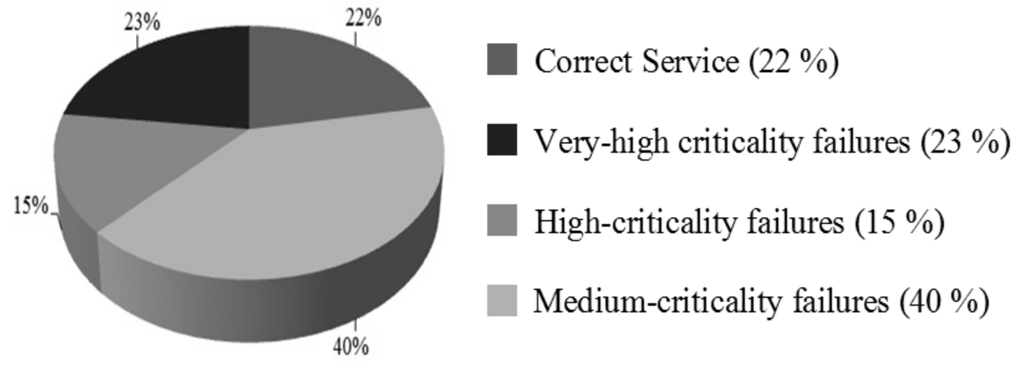

Figure 4. Results of the second campaign

4. Discussion

The effectiveness of tests performed in this work can be demonstrated by comparing the results of the first and second campaigns with the results from regression test obtained prior to this work, as reported in Fig. 5 (more details can be seen in [FADEL et al. 2011]). The regression tests performed previously resulted in 206 failures during one-and-a-half year testing (using the 16 OLT software versions). A test robot developed at CPqD was employed to perform regression testing of 850 test cases.

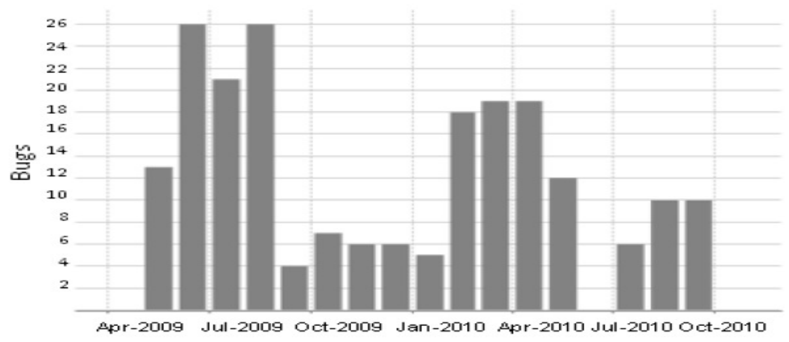

Figure 5. Regression test failures detected prior to this work

Observing the chart presented in Fig. 5, one can see that during the period of deployment of the robot (June-September, 2009) a large number of faults were found.

Another period when the number of faults has increased was the one between March and June 2010, when new critical features were implemented and delivered.

While the tests performed during one-and-a-half year found 206 failures in 16 software versions, a five-day test performed in the first campaign of this work uncovered 47 failures. Furthermore, 31 failures were found in just one day of test during the second campaign. In both cases the tests were applied to only one software version. Clearly, the technique employed in this work increased the probability of disclosing faults in GPON. By automating some test activities, it was possible to obtain more accurate results in less time.

The first campaign is an automated fault injection, based on test cases created from the OLT software state machine. Failures are emulations of physical faults that automatically break down the communication between GPON network devices. The use of automatic validation through fault injection in the state machines brought improvements to the software development and debugging processes. Without this technique, the software development team would have more difficulty in reproducing failures. Moreover, with automatic validation it was easier to trace the faults, since the developer had more control over the repeatability of the experiments.

The second campaign aims at validating the OLT software and it is also based on fault injection, but the target is the OMCI protocol. In this case, automation using the ConData tool has only been used for the generation of test cases based on state machines. The fault injection occurred manually, i.e. by manipulating the OMCI code in order to corrupt some messages.

If the faults identified in the first campaign affect an ONU, a telecom operator must dispatch a technician to the user's premises and reconfigure the ONU. In this case, the OLT may lose control of these ONU's and remote configuration becomes no longer possible. Another potentially disastrous scenario that could occur as a result of the disclosed faults is the need of reconfiguration of all ONUs, by the operator, due to the OLT breakdown. This recovery would require a cold restart of the central station. Clearly, this could affect all network users, who would have their service discontinued until the full recovery of the OLT.

5. Conclusions

This work showed that testing with fault injection using state machine model in a planned and documented way improved system quality. As these techniques have not been applied previously to GPON, the effort to implement these activities was significant. Mainly in the first campaign when the test robot was needed, 160 hours were spent, on average, to readapt the robot for this campaign. However, all the investment was rewarded by the results (number of faults found) and the possibility of rerunning these tests in other stages of system development.

State-machine models played a pivotal role in this work, as the same approach was applied to both campaigns and the results were more than satisfactory. They guided the fault injection procedure by determining the more suitable places to inject the faults. They were also used in order to create test cases. In addition, the test coverage was substantially improved, since state transitions were systematically exercised, a fact that was not possible without this model.

Although the two campaigns had the same objective, the second differs from the first one by adhering to CoFI, a well-defined methodology that supports the preparation, the execution and the reporting of the campaign results. The execution of the second campaign was not automated as the first one. In addition, the OMCI protocol provided the rules for the correct operation of the component under test, thus guiding the validation analysis.

The validation by automated fault injection (as performed during the first campaign) does not replace, but rather complements the development and debugging processes applied to the system prior to this work. The technique used increased the probability of disclosing faults that were difficult to reproduce manually. The transitions from one state to another usually last only a few milliseconds, resulting in the best case scenario in inadequate manual testing coverage, if not rendering manual testing unfeasible at all. Therefore, the work of finding and exposing faults, as well as reproducing the campaigns was significantly facilitated.

Before this work commenced, previous testing approaches were not able to detect the faults reported in this research. The results of both test campaigns clearly showed that the system under consideration must still be improved in order to meet the high-dependability requirements of GPON systems.

References

Ambrosio, A., Martins, E., VIJAYKUMAR, N., and DE CARVALHO, S. (2006). A conformance testing process for space applications software services,. *Journal of Aerospace Computing, Information, and Communication*, 3(4):146–158.

Arlat, J., Aguera, M., Amat, L., Crouzet, Y., Fabre, J.-C., Laprie, J.-C., Martins, E., and Powell, D. (1990). Fault injection for dependability validation - a methodology and some applications,. *IEEE Trans. on Software Engineering*, 16(2):166–182.

CPqD (2013). http://www.cpqd.com.br/en/. Website.

FADEL, A., MORAES, R., and MARTINS, E. (2011). Automated validation of embedded optical network software. In *Proc. Fifth Latin-American Symposium on Dependable Computing - LADC*, pages 1–6, São José dos Campos, BR.

HSUEH, M.-C., TSAI, T. K., and IYER, R. K. (1997). Fault injection techniques and tools. *Computer*, 30(4):75 – 82.

ITU-T-G.984.1 (2008). Gigabit-capable passive optical networks (gpon): General characteristics. International Telecommunication Union.

ITU-T-G.984.4 (2008). Gigabit-capable passive optical networks (gpon): Onu management and control interface specification. International Telecommunication Union.

MARTINS, E., SABIÌO, S. B., and AMBRîSIO, A. M. (1999). Condata: a tool for automating specification-based test case generation for communication systems. *Software Quality Journal*, 8(4):303 – 320.

RAMASWANI, R. and SIVARAJAN, K. (2002). *Optical Networks: A Pratical Perspective*. Morgan Kaufmann Publ. Second Edition.