Architectural Design and Test Case Analysis of a Simulator for Failure Localization

by

Xiangzhu Lu

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2024

Author's Declaration

Abstract

Failure localization involves identifying the suspicious locations of failures by analyzing the reported alarms recorded in the OTN control plane. To expedite the development of failure localization algorithms and reduce costs, a simulator is essential to replicate alarm propagation behaviors across various scenarios. This thesis presents the design and implementation of a simulator comprising the following components: a rule database, a topology generator, a failure generator, and an alarm generator.

The topology generator produces network topologies to simulate various network conditions, while the failure generator generates simulated failures. Subsequently, the alarm generator utilizes the rule database to generate corresponding alarm data. The generated data structures include failures/alarms, alarm flows, alarm chains, and alarm correlation trees. Additionally, two post-processing methods are introduced to illustrate the derivation of new data structures from existing data.

To validate the accuracy of the simulator, five test cases are introduced, featuring different topology settings, varying numbers of failures, and including a specific scenario involving noisy alarms.

Acknowledgements

Firstly, I would like to express my heartfelt gratitude to my supervisor, Professor Pin-Han Ho, for his invaluable guidance and unwavering support throughout this research journey.

Then, I want to extend my sincere appreciation to my colleagues, Yan Jiao and Zening Li, whose kindly assistance greatly contributed to my research.

Lastly, I am deeply thankful to my parents for their boundless encouragement, understanding, and love, which have been my constant pillars of strength.

Dedication

This thesis is dedicated to the people who have supported and encouraged me throughout this journey.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the backbone of modern communication systems, telecommunications networks have undergone significant evolution in the past decades, driven by the increasing demand to handle large volumes of data at high speeds. Historically, telecommunications networks relied primarily on wired technologies, such as copper cables. However, inherent limitations in bandwidth, transmission distance, and susceptibility to interference make it challenging for these systems to keep pace with the burgeoning data traffic of contemporary networks. Fiber optical communication system, on the other hand, offers solutions that facilitate such expansions through the utilization of optical fiber technology. By transmitting data in the form of light signals, it delivers unprecedented capacity, speed, and reliability, mitigating many of the shortcomings of traditional wired networks.

To enable compatibility among components developed by different manufacturers, numerous standards have been developed. Optical Transport Network (OTN) is a industry standard introduced by ITU-T that provides an efficient way to transport, switch and multiplex different services onto a single high-capacity optical lightpath [4]. It is also known as a "digital wrapper" because it encapsulates frames of data from various clients, including IP, Ethernet, storage, digital video and SONET/SDH, into one container for transport across optical networks, as illustrated in Figure 1.1. Moreover, by adding forward error correction (FEC) overhead, OTN supports the detection and correction of errors in the optical link.

However, with the existence of factors such as equipment malfunction and environmental threats, even the most robust networks are not immune to occasional faults and/or interruptions. In the event of such challenges, the ability to localize the source of failures becomes paramount.

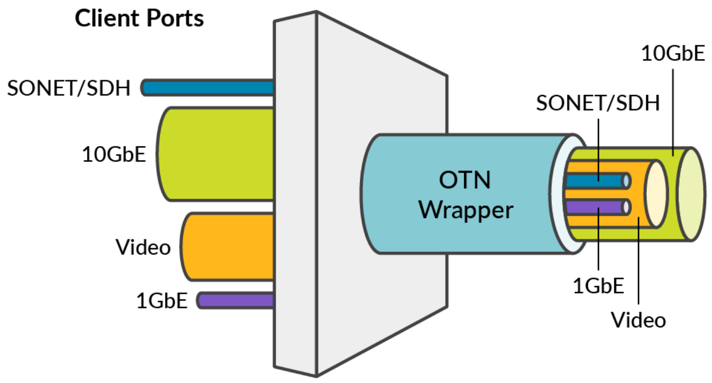


Figure 1.1: OTN Wrapper [1]

1.1 Motivation

Failure localization is the process of tracing signals within the network to determine or localize the initial malfunctioning node(s). A failure event can occur unexpectedly at any board or fiber segment, disrupting the optical signals traversing through it. Subsequently, the failure may trigger alarms that propagate across multiple boards nearby and/or those situated in geographically distant areas, depending on network topology and traffic distribution. However, only alarm events are recorded in the OTN control plane, and those events encompass not only the root alarms triggered directly by a failure, but also alarms that have propagated from other alarms.

To ensure the efficient troubleshooting and maintenance of the network, it is essential to pinpoint the root location of failure by analyzing the alarm records. Given the complex and dynamic nature of modern networks, having a simulator is essential. A simulator serves as a virtual environment where different network conditions and failure scenarios can be replicated in a controlled manner. Furthermore, it not only accurately reproduces real-world scenarios but also proactively generates potential failure scenarios.

1.2 Thesis Contribution

This thesis introduces an enhanced version of the OTN-based simulator, building upon the groundwork laid by its original version as discussed in [5]. The refined simulator incorporates the following advancements and functionalities:

- The refined simulator introduces dynamic topology functionality, mirroring real-world scenarios where optical networks evolve over time. Additionally, enhancements to the failure generator and alarm generator ensure seamless support for dynamic topology simulations.
- In contrast to the original version, which required manual coding of network topologies, the refined simulator now integrates a topology generator. This tool automates the production of diverse topologies and traffic scenarios of varying sizes and states, enhancing the simulator's versatility and usability.
- In the original version, the simulator operated at the node level, necessitating separate code blocks for each node type within the alarm generator. For instance, to locate the nearest board of type A downstream from location B, the simulator would first identify the node type containing a type A board, then ascertain the nearest such node, and finally determine the closest board to location B. In contrast, the refined simulator operates at the board level, enabling the reuse of the same board-search functions across all node and board types, thus enhancing modularity within the simulator.
- Additional functionalities have been incorporated into the alarm generator, comprising the following features:
  - In contrast to the original version, which utilized a fixed time step for every alarm pair, the refined simulator now incorporates a random propagation time between alarm pairs.
  - Each alarm now includes a unique Alarm ID, allowing for the distinction of alarm events with identical types and locations.
  - Noise alarms can now be introduced to the alarm set, facilitating testing of the failure localization methods' anti-noise capabilities.

1.3 Thesis Organization

The remainder of this thesis is structured into four chapters, outlined as follows:

Chapter 2 delves into the background information surrounding optical transport networks, elucidating their structures and primary components. Additionally, this chapter offers a comprehensive overview of key definitions, encompassing terms such as topology, node, board, lightpath, alarm and trail.

Chapter 3 delineates the architecture of the simulator, detailing its core components including the rule database, topology generator, failure generator, and alarm generator. Furthermore, this chapter introduces two post-processing methods utilized for feature extraction from the generated data.

Chapter 4 outlines four distinct test cases situated in various scenarios, comprising three for static topology and one for dynamic topology.

Chapter 5 summarizes the thesis and delineates its limitations, offering insights for prospective directions.

Chapter 2

Background

2.1 Overview of Optical Transport Network

Optical transmission network combines the benefits of synchronous optical networking and synchronous digital hierarchy (SONET/SDH) with the bandwidth expandability of dense wavelength division multiplex (DWDM) technology [6]. SONET/SDH are standardized network protocols that specify methods for multiplexing digital signals in order to transmit data over optical fiber networks [7]. Moreover, DWDM technology allows OTN to combine data signals from different sources onto a single pair of optical fiber, maximizing the utilization of network resources.

2.1.1 Hierarchical Structure of OTN Layers

G.709 defines a number of layers in the OTN hierarchy, which are shown in Figure 2.1. These layers include OPUk, ODUk, and OTUk layers, residing within the electrical/digital domain, whereas OCh, OMS, and OTS layers are in the optical/analog domain. To transport a client signal, the OTN encapsulates it through the following procedures [4]:

1. The OPUk encapsulates the client signal and adds overhead needed to perform rate adaptations.

2. The ODUk adds overhead to conduct tandem connection monitoring (TCM) and end-to-end section supervision.

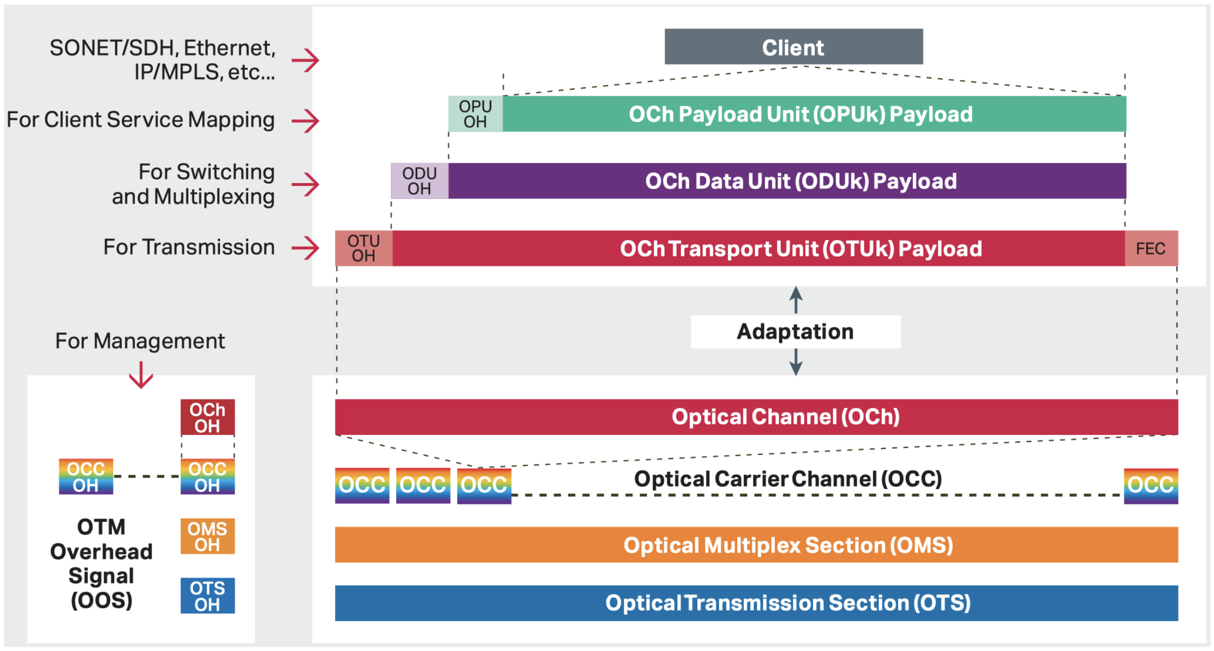

SONET/SDH, Ethernet, IP/MPLS, etc...
For Client Service Mapping
For Switching and Multiplexing
For Transmission
For Management
Client
OPU OH
OCh Payload Unit (OPUk) Payload
ODU OH
OCh Data Unit (ODUk) Payload
OTU OH
OCh Transport Unit (OTUk) Payload
FEC
Adaptation
OCh OH
OCC OH
OCC OH
OMS OH
OTS OH
OTM Overhead Signal (OOS)
Optical Channel (OCh)
OCC
Optical Carrier Channel (OCC)
Optical Multiplex Section (OMS)
Optical Transmission Section (OTS)

Figure 2.1: OTN Hierarchy [2]

3. The OTUk adds overhead to supervise and condition the signal for transport between 3R (Re-amplification, Re-shaping, and Re-timing) regeneration points. Additionally, a forward error correction (FEC) code is added to monitor error control and a frame alignment signal (FAS) is added to support frame synchronization.

4. Now the frame is fully formatted and can be transmitted over a wavelength, which constitutes OCh. The OTN frame structure is given in Figure 2.2.

5. The OMS aggregates group of optical channels onto a single wavelength-division multiplexing channel. It manages fiber links between optical multiplexer and switches.

6. The OTS represents the physical layer of the optical network, which manages fiber links between optical components such as optical amplifiers.

To reduce complexity and provide a deeper exploration, the simulator is designed to focus on the optical layer only.

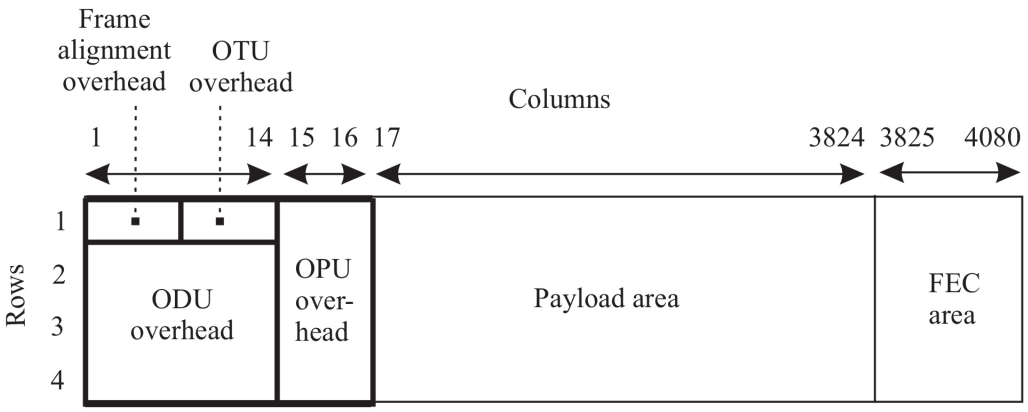


Figure 2.2: OTN Frame Structure [3]

2.1.2 Key Equipment in OTN

Various types of OTN equipment are deployed according to the standards. Below are the most prevalent types and their main functionalities:

- Optical Transponder: converts the digital signals into optical signals for transmission.
- Optical Add/Drop Multiplexer (OADM): selectively drops/inserts optical signals from/into the fiber.
- Optical Amplifiers: boosts optical signals to extend transmission distances.
- Optical Regenerators: reconditions the received degraded optical signal.

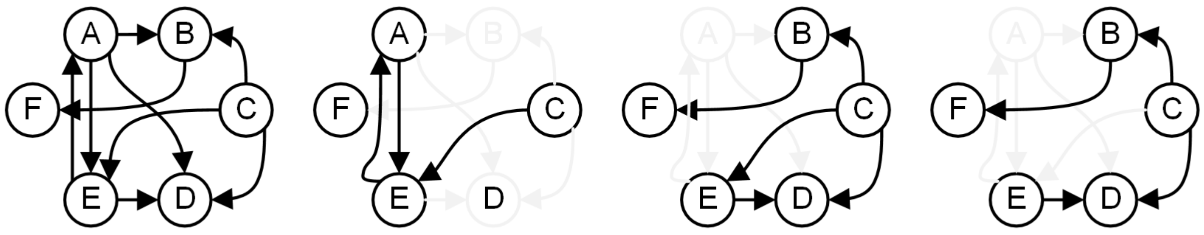
2.2 Key Concepts and Definitions

This section provides a comprehensive overview of the main components and concepts defined in the simulator.

2.2.1 Topology

Topology is the physical arrangement of nodes and connections in the network. In the context of alarm propagation, if alarm A triggers alarm B, then their locations must be connected in the topology, either directly or through a path.

A topology can be static or dynamic. A **static topology** assumes that the network components (nodes and connections) remain unchanged over time. On the contrary, a **dynamic topology** assumes that the nodes and connections can be added or removed to form different network states. In the simulator, a **network state** is represented by a sub-topology of the entire network. To be more specific, the nodes and connections in a network state constitute a subset of those present in the entire network. Figure 2.3 is an example of dynamic topology. Figure 2.3a is the entire network, and Figures 2.3b, 2.3c and 2.3d are three different network states. A static topology has only one network state.



(a) (b) (c) (d)

Figure 2.3: Example of Dynamic Topology

2.2.2 Board

A **board** serves as the smallest unit in the topology. There are five types of boards in the simulator: FIU, OA, OM, OD and OTU. Fiber interface unit (FIU) serves as an intermediary between fiber and another node. A **connection** exists between two nodes when their FIUs are linked by two fibers in opposite directions. Optical multiplexer (OM) and optical demultiplexer (OD) boards facilitate the addition/dropping of optical signals, while the optical amplifier (OA) boosts signal strength. Lastly, the optical transponder unit (OTU) serves as the endpoint of a lightpath, where each OTU corresponds to at most one lightpath.

Similarly, a **fiber** connects two boards together. It is essential to recognize that fibers possess directionality. For example, an OA_FIU fiber denotes a fiber from an OA board to a FIU board. There are eight types of fiber: FIU_FIU, OA_FIU, FIU_OA, OA_OD, OD_OM, OM_OA, OD_OTU and OTU_OM.

2.2.3 Node

A **node**, comprising boards and fibers, represents an equipment in the network. There are two types of nodes involve in the simulator: reconfigurable optical add-drop multiplexer (ROADM) and optical line amplifier (OLA).

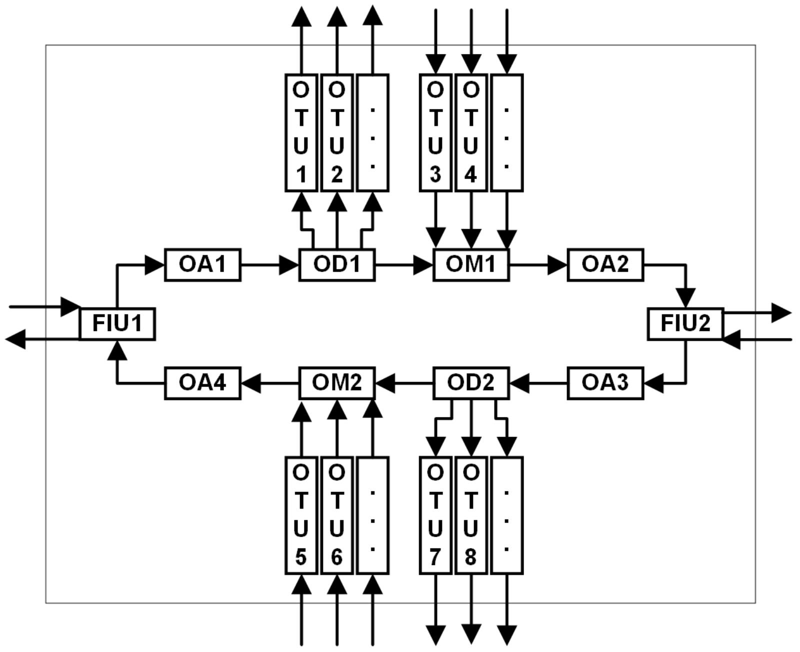


Figure 2.4: ROADM Structure

Their structures are shown in Figures 2.4 and 2.5. In the case of OLA, all boards and fibers have a fixed count and layout. However, the number of OTUs in a ROADM depends on the quantity of lightpaths originating from and terminating at it.

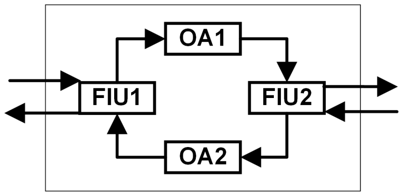


Figure 2.5: OLA Structure

2.2.4 Lightpath

A **lightpath** is a directional path between two OTUs in the topology. Below are the three scenarios in which a node can possess a lightpath, along with their corresponding board-level paths:

- The lightpath originates from a node:
  - ROADM: OTU → OM → OA → FIU → . . .
  - OLA: there will be no lightpath originating from it.
- The lightpath terminates at a node:
  - ROADM: . . . → FIU → OA → OD → OTU
  - OLA: there will be no lightpath terminating at it.
- The lightpath traverses through a node:
  - ROADM: . . . → FIU → OA → OD → OM → OA → FIU → . . .
  - OLA: . . . → FIU → OA → FIU → . . .

We can observe that the board-level path has a fixed pattern. Therefore, given the initiating and ending OTUs, the board-level lightpath can be derived from the node-level lightpath. In the simulator, a lightpath is represented by a list of boards. However, to enhance clarity and simplicity, this thesis will depict lightpaths at the node level in the figures.

A **traffic** comprises a list of lightpaths, and a network state encompasses only one traffic, including all the active lightpaths transporting signals in the network state. Figure 2.6 is an example of traffic in a network state. There are three lightpaths in the traffic: C → B → F, C → E → D, C → D.

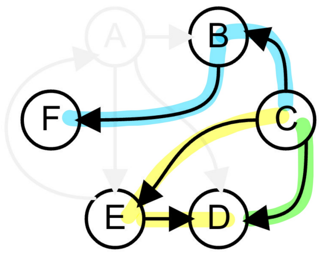


Figure 2.6: Example of Traffic

2.2.5 Alarm

Failure is an exceptional event that can trigger alarms in the OTN. It can occur on any board or fiber in the topology. **Alarm** is an event triggered by an alarm/failure, which can only be located on a board. When an alarm directly results from a failure, it is considered the **root alarm** of that failure. It is worth noting that a single failure can generate multiple root alarms.

An alarm/failure possesses the following properties in the simulator:

- *Alarm type* refers to the specific category of the alarm/failure. The simulator defines a total of 25 alarm types, and 2 failure types.
- *Location* denotes the board/fiber where the alarm/failure occurs.
- *Time* denotes the moment when the alarm/failure happens, measured in seconds.
- An unique *ID* is assigned to every alarm and failure.
- *Failure ID* is the ID of the failure which leads to the alarm.
- *Is root* indicates whether the alarm/failure is a root alarm.
- To test the anti-noise capability of the failure localization method, random noisy alarms can be added to the alarm set. *Is noisy* indicates whether the alarm is a noisy alarm.

Below is an example of a failure and its corresponding alarms:

Alarm type	Location	Time	ID	Failure ID	Is root	Is noisy
board faulty	ROADM5-OD1	06:00:00	40	40	false	false
OCh_LOS_P	ROADM5-OTU1	06:00:10	264	40	true	false
OCh_LOS_P	ROADM7-OTU93	06:00:04	265	40	true	false
OMS_A_P	ROADM5-OD1	06:00:07	266	40	true	false
OCh_A_P	ROADM5-OTU1	06:00:14	267	40	false	false
OCh_A_P	ROADM7-OTU93	06:00:15	268	40	false	false

Table 2.1: Example of Failure and Corresponding Alarms

To demonstrate the causal relationship within the failure and alarm set, the concept of alarm flow is introduced. An **alarm flow** consist of either a failure and an alarm or two alarms, where A → B indicates that failure/alarm A triggers alarm B. The aggregation of all alarms and alarm flows of one failure forms an **alarm correlation tree**, as given in the Figure 2.7, using the same failure example in Table 2.1.

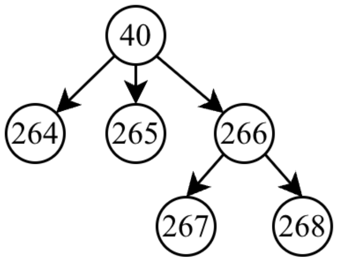


Figure 2.7: Example of Alarm Correlation Tree

Each leaf alarm in the alarm correlation tree is associated with an **alarm chain**, which is a path from failure to respective leaf alarm. In Figure 2.7, there are four alarm chains: 40 → 264, 40 → 265, 40 → 266 → 267 and 40 → 266 → 268.

2.2.6 Trail

There are three trails defined in the simulator: OTS, OMS, and OCh trails, corresponding to the three layers of the OTN. The **OTS trail** is located between any two connected boards in the topology, meaning each fiber is an OTS trail. The **OMS trail** is located between the OM and OD of two nodes, where a directional path exists between them and no other OM or OD is located in this path. The **OCh trail** is a span on a single lightpath, located between two OTUs.

Chapter 3

Simulator Architecture

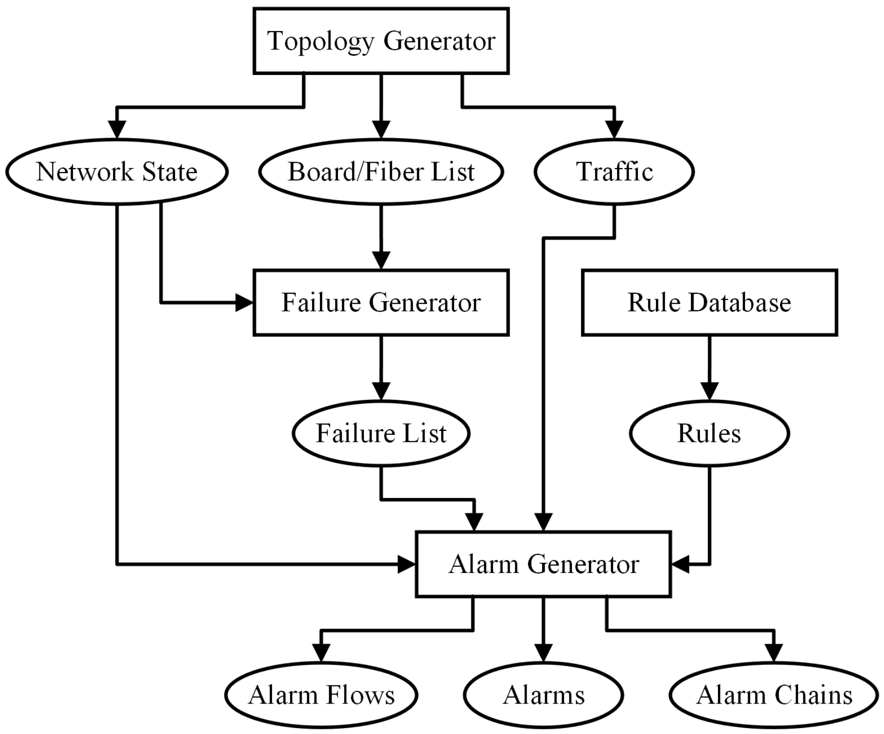


Figure 3.1: Simulator Architecture

A simplified architecture of the simulator is depicted in Figure 3.1, where static topology is employed. In the case of dynamic topology, multiple sets of data including *Network State*, *Board/Fiber Lists*, and *Traffic* will be generated. For each set, the simulator will apply the failure generator and alarm generator. Below are the steps the simulator will execute:

1. Firstly, the topology generator will create a topology according to the specified requirements. Then for each network state, a traffic will be generated based on the required number of lightpaths. Board/fiber list is the list of all boards and fibers in each network state.

2. Given all boards and fibers, the failure generator will randomly choose failure locations.

3. Finally, the alarm generator can generate alarm data for each failure based on the rules provided by rule database.

3.1 Rule Database

The rule database is stored in MySQL and accessed by the simulator through the PyMySQL library. It provides information about the propagation behavior of each alarm type and failure type. To make a query, the simulator needs to input "Board" and "Receive Detect Event", after which MySQL will output the values of "Output Board", "Output Type", and "Output". Here is an explanation of each attribute:

- *Board* is the location type of the input failure/alarm. It is required because the occurrence of an alarm type on different location types may result in different outcomes. For example, if an OMS_LOS_A alarm is detected by an OD, it will propagate an OCh_LOS_P alarm to the OTU. However, if the same alarm type is detected by an OM, it will trigger an OMS_SSF_E alarm on the OD.

- *Receive Detect Event* refers to the type of the input failure/alarm.

- *Output Board* is the location type of the alarm that will be triggered.

- *Output Type* specifies the direction of propagation, including transmit downstream, transmit upstream and locally report. If the output type is transmit downstream, the alarm propagates along the direction of fibers. Conversely, if the output type is transmit upstream, the alarm propagates in the opposite direction of fibers. Locally report means that the alarm/failure triggers another alarm at the same location. Visualization of each output type is provided in Figure 3.2, where an vertex represents a board.

- *Output* is the type of the alarm that will be triggered.

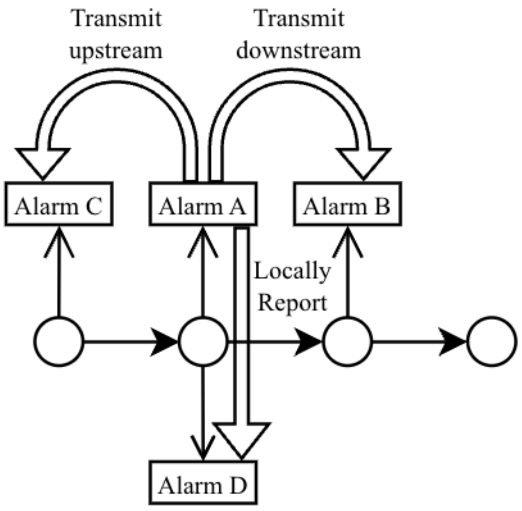


Figure 3.2: Different Output Type Behaviour on Board Level

Table 3.1 are two sample rules in the rule database. The first rule is a failure → alarm rule. If OA_OD detects a fiber cut failure, it will transmit an OMS_LOS_A alarm along the direction of fibers to the next OD. The second rule is a alarm → alarm rule. After an OTS_LOS_B is reported on FIU, it will trigger two alarms: an OTS_LOS_A on FIU and an ONS_SSF_B on OD.

Board	Receive Detect Event	Output Board	Output Type	Output
OA_OD	fiber cut	OD	transmit downstream	OMS_LOS_A
FIU	OTS_LOS_B	FIU	transmit downstream	OTS_LOS_A
		OD	transmit downstream	OMS_SSF_B

Table 3.1: Example of Rules

There are a total of 2 failure types and 25 alarm types: fiber cut, board faulty, OTS_LOS_A, OTS_LOS_B, OTS_LOS_C, OTS_LOS_P, OTS_LOS_O, OMS_LOS_A, OCh_LOS_P, OTS_A_P, OMS_A_P, OCh_A_P, OTS_BDI_A, OTS_BDI, OMS_BDI, OMS_SSF, OMS_SSF_B, OMS_SSF_P, OTS_PMI, OMS_BDI_O, OMS_SSF_O, OMS_SSF_J, OMS_SSF_A, OMS_SSF_C, OMS_SSF_E, OMS_SSF_F, OTS_LOS_O, which can be classified into four categories: failure, OTS, OMS and OCh.

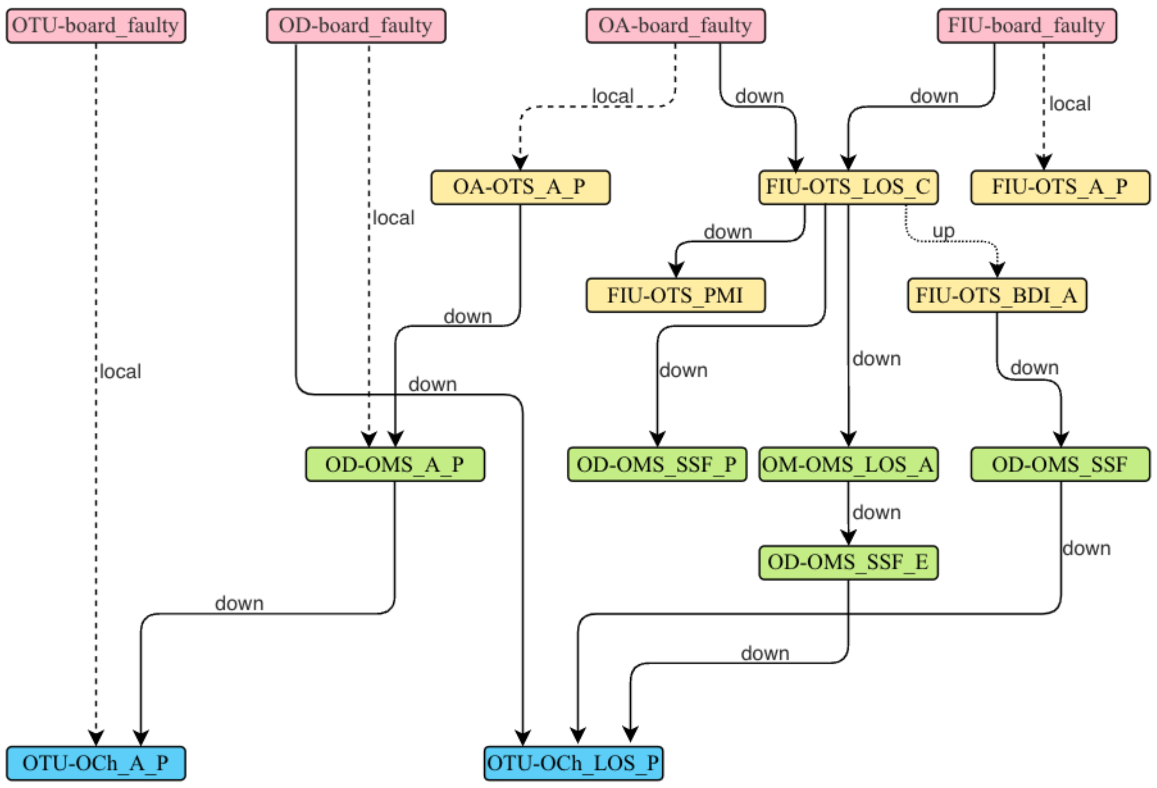


Figure 3.3: A Subgraph of Rule Database

Figure 3.3 is a subgraph of rule database, where a rule is represented by *Board-Receive Detect Event* $\xrightarrow{Output\ Type}$ *Output Board-Output*. Let a (location type, failure/alarm type) pair be a rule event in the rule database. It is noticeable that a rule event can trigger multiple other rule events, and conversely, multiple rule events can trigger the same rule event.

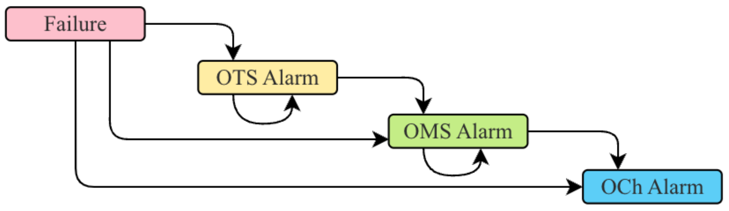


Figure 3.4: Hierarchy of Rule Events

Furthermore, there is a hierarchy in the rule events, as shown in Figure 3.4. At the highest level, failures have the ability to trigger all other alarm types. OTS alarms and OMS alarms can trigger all alarm types on the same or lower level than themselves. OCh alarms are unable to trigger any alarms, including those at the same level.

3.2 Topology Generator

3.2.1 Static Topology Generator

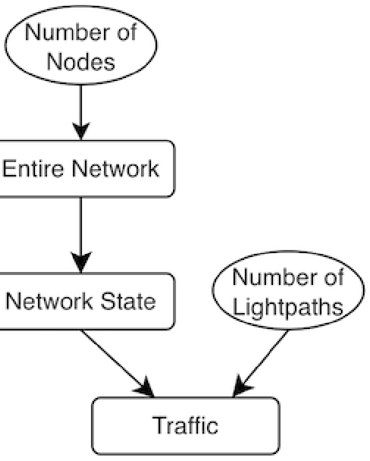


Figure 3.5: Flowchart of Static Topology Generator

To generate a static topology, two parameters are required: *Number of Nodes*, which determines the size of the topology, and *Number of lightpaths*, representing the traffic volume. The flowchart of the static topology generator is shown in Figure 3.5, and the step-by-step procedures are outlined below:

1. A random node-level directed graph is generated with the help of NetworkX package. The ratio of ROADMs is randomly selected between 0.85 and 0.95. Node A is connected to node B (A $\rightarrow$ B) if FIU2 of node A and FIU1 of node B are linked by two opposing fibers.

2. Since static topology has only one network state, there is no necessity to select a subset of nodes and connections from the entire network. Therefore, the network state is regarded as identical to the entire network.

3. Next, a number of lightpaths will be generated based on the network state, where the pseudo-code is provided in Algorithm 1. The number of available OTUs should be significantly larger than the number of required lightpaths.

Algorithm 1 Generate Lightpaths

```
Input: network state $NS$, number of lightpath $N_L$
Output: $N_L$ unique lightpaths
  $O$ = all OTUs in $NS$
  $L = \emptyset$
  for $k = 1$ to $N_L$ do
      $b_s$ = randomly select an OTU from $O$
      $n$ = node location of $b_s$
      $l_n$ = node-level lightpath with start node $n$
      while $\exists$ a neighbour $r$ of $n$ do
          add $r$ to $l_n$
          $n = r$
      end while
      $b_e$ = randomly select an OTU from $N$
      $l_b$ = board-level lightpath given $l_n, b_s, b_e$
      add $l_b$ to $L$
      remove $b_s, b_e$ from $O$
  end for
  return $L$
```

3.2.2 Dynamic Topology Generator

The process of generating a dynamic topology is similar to that of a static topology:

1. A random node-level directed graph is generated to form the entire network.

2. To mimic the real-life scenario where lightpaths are added and removed over time, we first establish the list of traffics, and then construct the network state based on each traffic. The pseudo-code is provided in Algorithm 2 and the flowchart is given in Figure 3.6.

Algorithm 2 Generate Traffics

```
Input: entire network $EN$, number of traffic $N_T$, number of lightpaths for each traffic
    $N_1, N_2, \cdots N_{N_T}$
Output: $N_T$ unique traffics $T_1, T_2, \cdots, T_{N_T}$ and $N_T$ network states $NS_1, NS_2, \cdots, NS_{N_T}$
  $T_1$ = Generate Lightpaths($EN$, $N_1$)
  $NS_1$ = all nodes and connections used in $T_1$
  for $k = 2$ to $N_T$ do
      $T_k$ = randomly remove some lightpaths from $T_{k-1}$
      $NS_k$ = all nodes and connections used in $T_k$
      $NS_k$ = randomly add some nodes and connections from $EN - NS_k$
      $T_k = T_k$+ Generate Lightpaths($NS_k$, $N_k - |T_k|$)
  end for
```

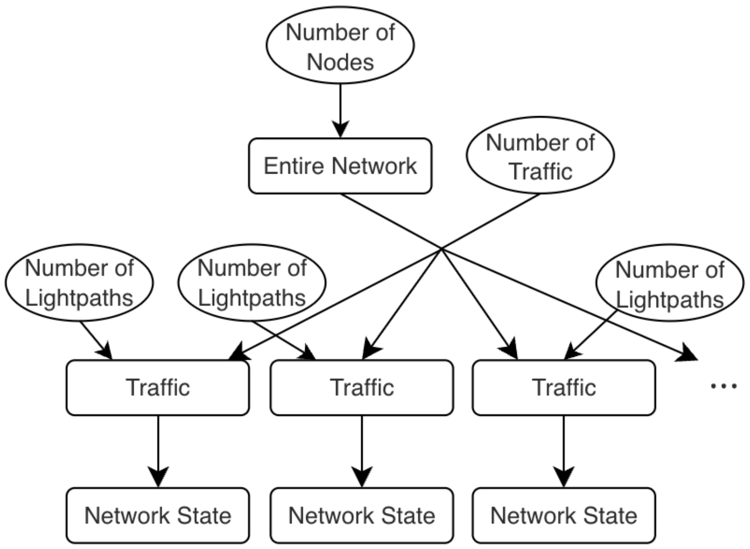


Figure 3.6: Flowchart of Dynamic Topology Generator

3.3 Failure Generator

The simulator can generate failures based on either a list of location types or a list of locations:

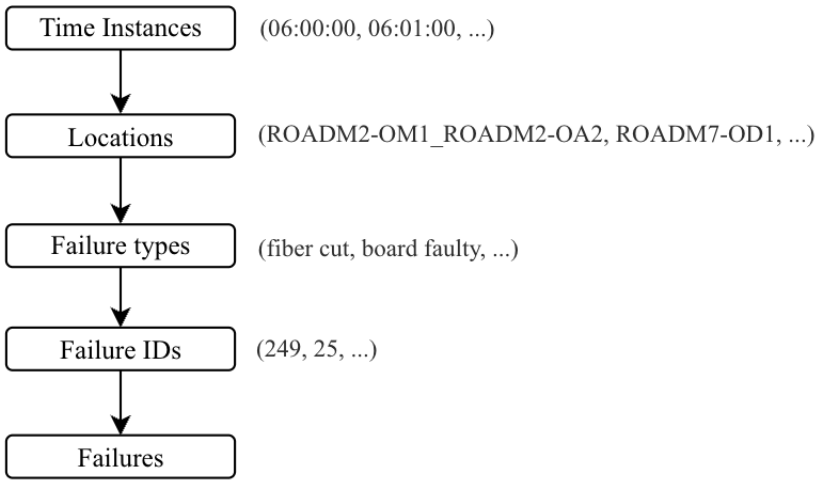


Figure 3.7: Flowchart of Failure Generator

1. To produce a list of failures, the simulator will first generate a list of time instances, indicating when each failure occurs. By default, we assume that each failure occurs at the beginning of each minute. Moreover, the number of failures per minute should be specified with the default set to 1.
2. Then, the simulator will select a list of locations if not provided. With a location type, the generator will first determine the eligible node type. For example, if the location type is OM, then only ROADMs are valid. Subsequently, a random node with the required node type(s) is selected from the network state, and a random location with the required location type is chosen from the node.
3. After determining the location, the failure type can be derived. Currently, there are only two failure types: fiber cut and board faulty, which occur on fiber and board correspondingly.
4. Since each location can fail only once, the ID of the failure is set to the ID of the location.
5. Now that all the attributes are prepared, the list of failures can be constructed.

3.4 Alarm Generator

The architecture of the alarm generator comprises two buffers, one event processor and two containers. Buffer A stores all the failure events that haven't been processed, while buffer B stores all the events resulting from a single failure event. The event processor takes an event as input and outputs all the alarms triggered by the event. The two containers store all the alarms and alarm flows generated. The steps to generate alarms given a list of failures are outlined below:

Initially, buffer A contains all the failures, while buffer B is empty.

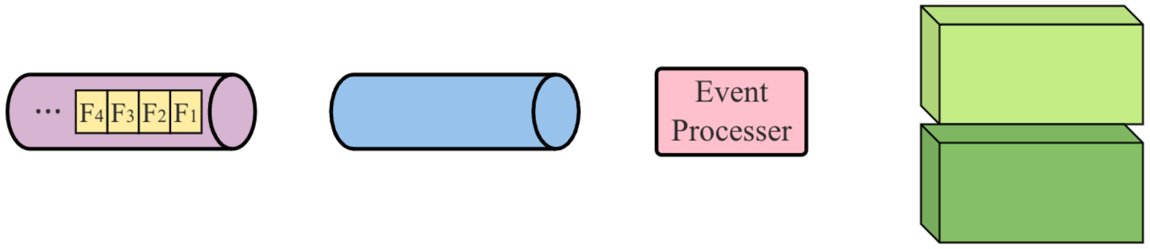


Figure 3.8: Alarm Generator (Status 1)

Then, the first failure F_1 in buffer A is moved to buffer B for processing.

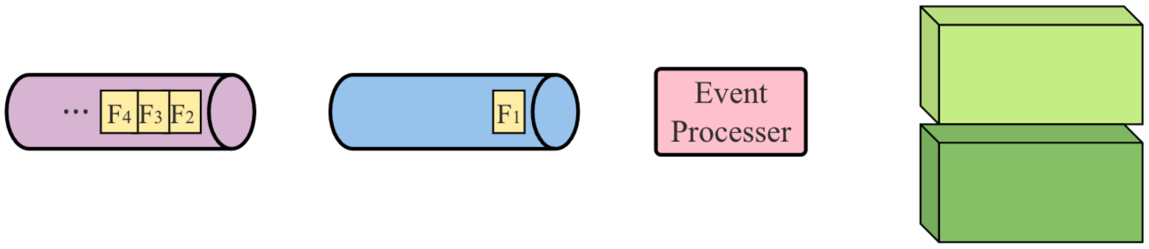


Figure 3.9: Alarm Generator (Status 2)

F_1 is input to the event processor, which then returns a list of alarms A_1, A_2, which are directly triggered by F_1.

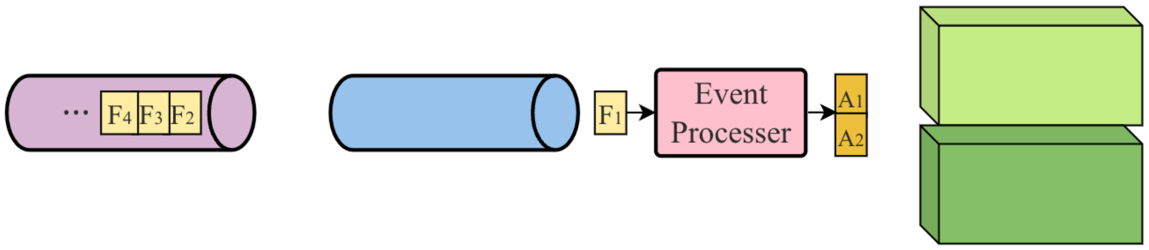


Figure 3.10: Alarm Generator (Status 3)

The alarms A_1 and A_2, along with their corresponding alarm flows $F_1 \to A_1$ and $F_1 \to A_2$, are added to the containers. Additionally, a copy of each alarm is appended to buffer B, as they may trigger further alarms.

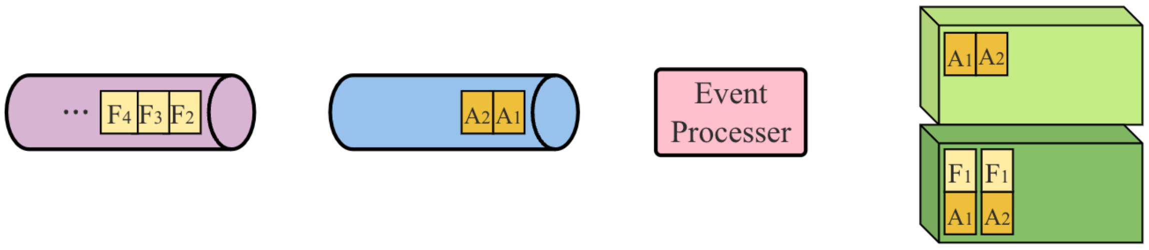


Figure 3.11: Alarm Generator (Status 4)

Then, the next event in buffer B is passed to the event processor.

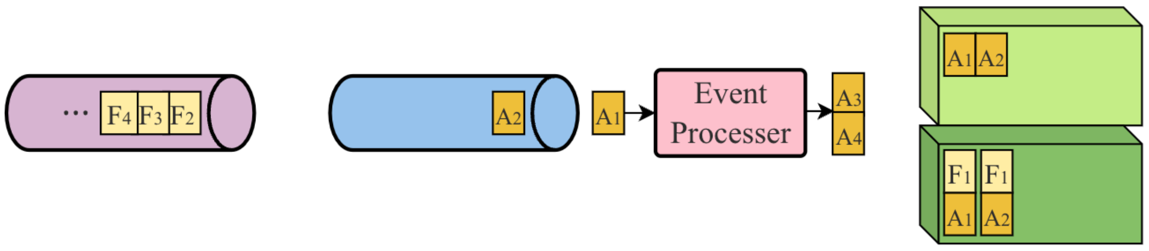


Figure 3.12: Alarm Generator (Status 5)

And the new outputs are added to both the containers and buffer B.

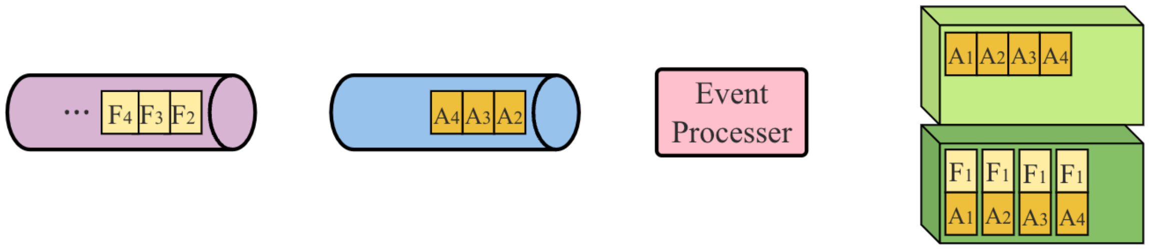


Figure 3.13: Alarm Generator (Status 6)

If no alarms are returned by the event processor, nothing will be added to the containers and buffer B.

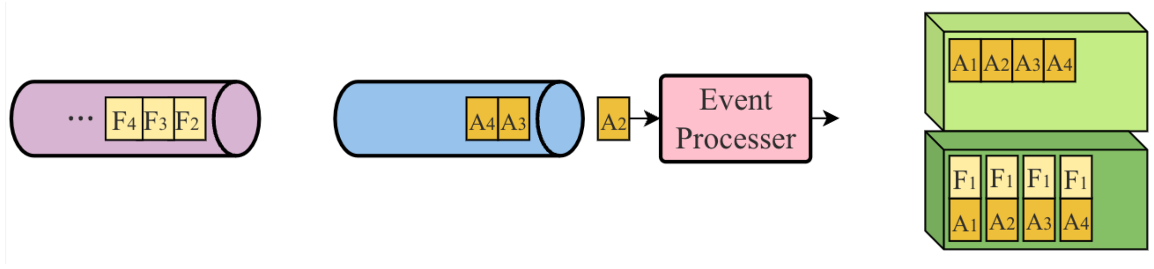


Figure 3.14: Alarm Generator (Status 7)

Keep processing the events until buffer B is empty. Now we have all the alarms and alarm flows that result from F_1.

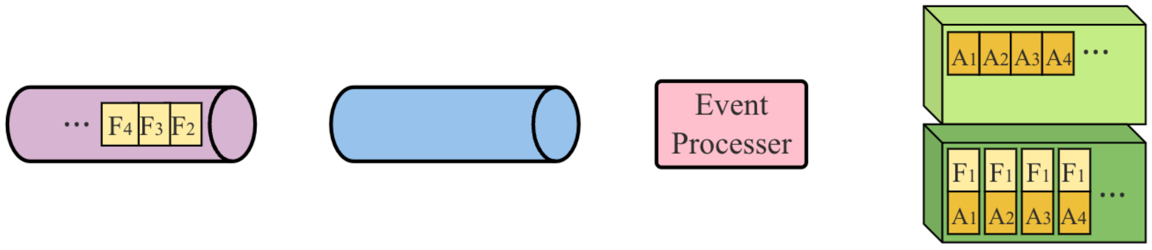


Figure 3.15: Alarm Generator (Status 8)

Then the next failure in buffer A is moved to buffer B, and the aforementioned steps are reiterated.

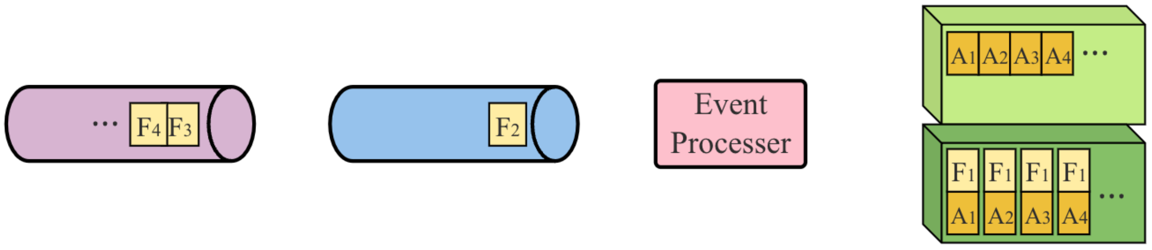


Figure 3.16: Alarm Generator (Status 9)

The alarm generation process terminates until buffer A and B are both empty. At this point, the containers hold all alarms and alarm flows generated from the list of failures.

3.4.1 Event Processor

This section illustrates how the event processor produces alarms by utilizing the rule database. Upon receiving an event e_i as input, the event processor will first extract the board type from its location. Subsequently, it will input this information, along with e_i's alarm type, into the rule database. Then the rule database will return a list of tuple (*Output Board*, *Output Type*, *Output*), as elaborated in Section 3.1.

For each tuple retrieved, the event processor will generate a list of alarm events, as depicted in Figure 3.17. The number of alarm events produced corresponds to the number of locations that meet the specified criteria. For example, if the retrieved tuple is (Transmit downstream, OTU, OCh_LOS_P), and there are two OTUs located downstream of e_i then two alarm events will be created, each corresponding to one of these locations. The subsequent content in this section expounds on the procedure for generating each attribute in the alarm event.

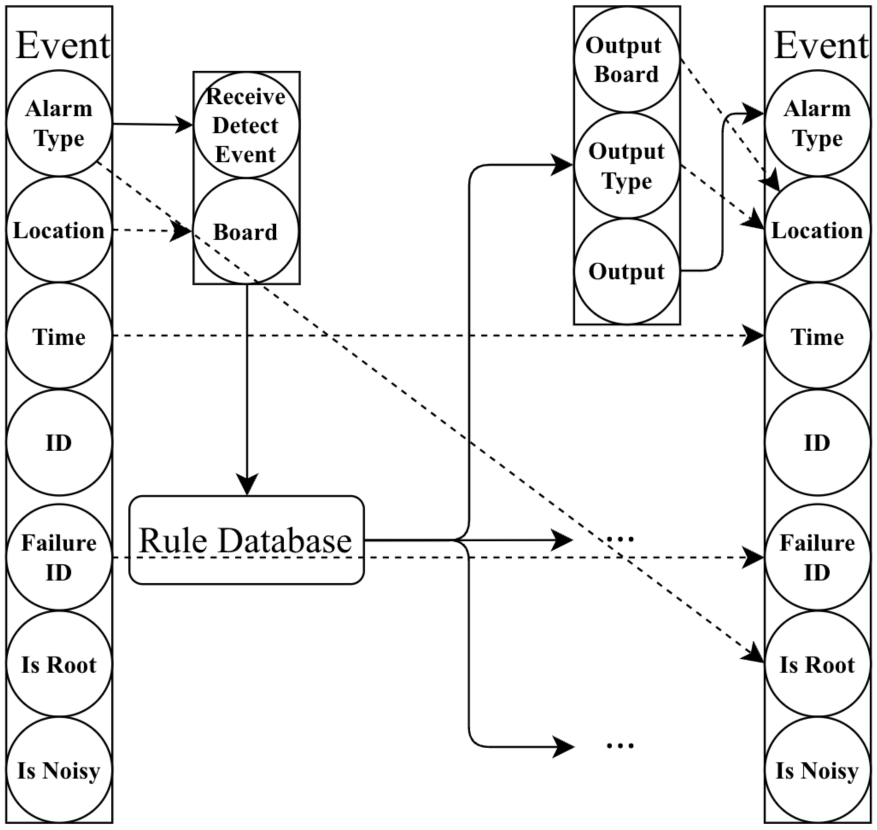


Figure 3.17: Event Processor Architecture

Alarm type

The *Alarm type* will match the value of the *Output* specified in the tuple.

Location

The *location* is determined by all the values within the tuple, alongside the board-level network state. There are three cases:

- Case 1: *Output Type* is "to board".
  A single alarm event will be generated, and its *location* will be identical to that of e_i.

- Case 2: *Output Type* is “transmit downstream/upstream”, and *Output* is an OTS or OMS alarm.
  The processor will search for all boards in the network state that fulfill the following criteria:

  - The board possesses a board type matching *Output Board*.
  - There exists a path between e_i and the board in the direction indicated by the *Output Type*.
  - No other boards with the *Output Board* type are located along the path.

  To optimize runtime, the processor will not manually search through all boards in the network state. Instead, it will conduct a breadth-first search starting from the location of e_i.

  A queue is utilized, where boards are dequeued sequentially at each step, and their neighbors are inspected to determine whether they should be enqueued or discarded, according to Algorithm 3. It is worth noting that this algorithm only addresses the scenario where the *Output Type* is “transmit downstream.” In the event of “transmit upstream,” the predecessors of each board will be explored instead.

Algorithm 3 Search Boards

```
Input: network state $NS$, $e_i$'s location $b_i$
Output: $boardlist$
  $Q$ = Queue($b_i$)
  $boardlist = \emptyset$
  while $Q$ is not empty do
      $newQ$ = Queue()
      while $Q$ is not empty do
          $b = Q$.dequeue()
          for $n$ in $b$.successors do
              if $n.type$ == $Output\ Board$ then
                  $boardlist$.add($n$)
              else
                  $newQ$.enqueue($n$)
              end if
          end for
      end while
      $Q = newQ$
  end while
```

Figure 3.18 offers a graphical illustration example to elucidate this process, while the network state is depicted in Figures 3.19 and 3.20. In the figures, "R" denotes "ROADM" and "O" denotes "OLA". Certain boards in the board-level figure have been omitted. In this example, we assume the rule database outputs (OM, transmit downstream, OMS_LOS_A), and e_i is on ROADM1-FIU1.

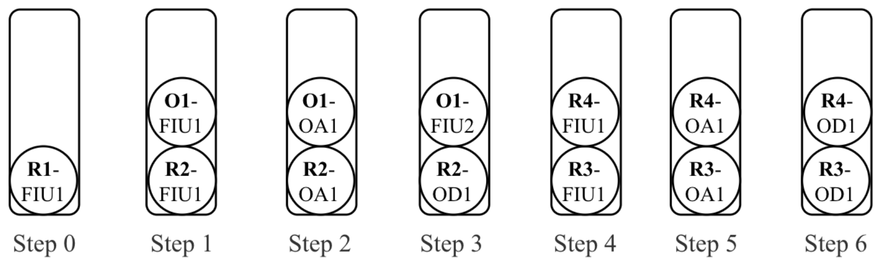


Figure 3.18: Example of Event Processor

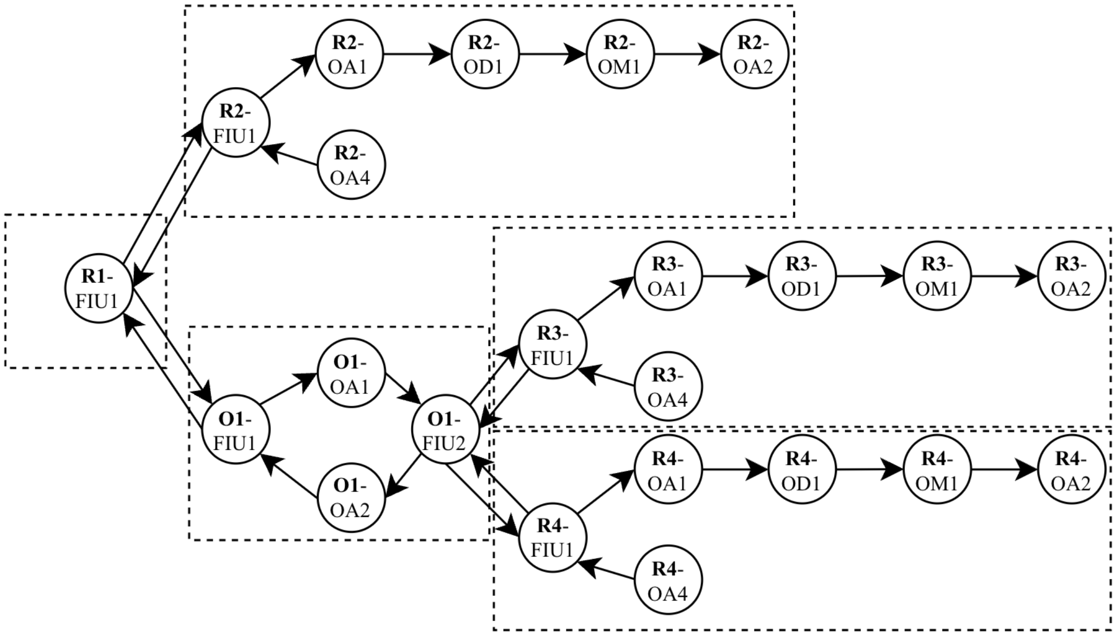


Figure 3.19: Board-level Network State of Event Processor Example

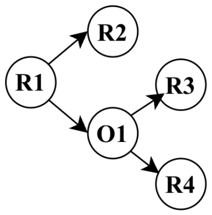


Figure 3.20: Node-level Network State of Event Processor Example

- Case 3: *Output Type* is "transmit downstream", and *Output* is an OCh alarm.
  The processor will iterate through all lightpaths, specifically identifying those that traverse the faulty board and subsequently pass through the location of e_i. It will then return the endpoints (OTUs) corresponding to these identified lightpaths. In the context of OCh alarm, there is no scenario where the *Output Type* is "transmit upstream".

Time

To simulate the real-life scenario, a random integer will be selected to represent the time required for an event to trigger another. To be more specific, for any event e_o triggered by e_i, its occurrence time will be the time of e_i plus a randomly generated duration ranging from 1 to 10 seconds.

ID

The alarm container will keep track of the number of alarms it receives and increment the ID by 1 each time a new alarm is inputted.

Failure ID

The alarm event will inherit the failure ID from the event that triggers it.

Is root

This attribute will be set to true if the alarm type of the parent event e_i indicates a failure; otherwise, it will be set to false.

Is noisy

By default, this attribute is set to false unless explicitly modified.

3.5 Post-processing for Feature Extraction

In order to apply various failure localization methods, additional data processing may be required to derive valuable information from the topology and/or alarm data. This section delves into two post-processing procedures to extract new features from the data.

3.5.1 Alarm Chain

The alarm chain, as introduced in Section 2.2.5, is a sequence of events denoted as $f_1 \rightarrow e_2 \rightarrow \cdots \rightarrow e_k$, where each pair of consecutive events constitutes an alarm flow. Here, f_1 represents the initial failure event, while e_k denotes the leaf alarm event. To construct the alarm chains for a specific failure, all the leaf alarms from the alarm set will be selected, and the chain should be developed in reverse order, tracing back from the leaf alarms to the originating failure event in the alarm correlation tree.

Given an alarm and an alarm flow set, the first step is to eliminate all noisy alarms from the alarm set, utilizing the *Is noisy* field in each alarm object. Subsequently, the alarms and alarm flows are grouped based on their *Failure ID*, and for each group, the following steps will be executed:

1. Input alarms and alarm flows into ArangoDB, a graph database system, to construct an alarm correlation tree, where each vertex represents an alarm/failure and each edge represents an alarm flow.
2. Locate all the leaf alarms in the tree by isolating all the vertices that have no outgoing edges.
3. For each leaf alarm e_k, trace back along its inward edge to identify the preceding event e_{k-1} such that $e_{k-1} \rightarrow e_k$. This edge is distinct because each alarm is uniquely triggered by an event.
4. Repeat step 3 for alarms $e_{k-1}, e_{k-2}, \cdots$ until reaching the failure f_1.
5. Then $f_1 \rightarrow e_2 \rightarrow \cdots \rightarrow e_k$ forms the alarm chain for leaf alarm e_k. The number of alarm chains for a failure equals the number of leaf alarms in its alarm correlation tree.

3.5.2 Distance

To localize the failure given a list of alarms recorded in the OTN control plane, one approach is to identify the dependency between each pair of collected alarms, thus enabling inference of the failure event with reduced complexity [8].

To determine whether there is an alarm flow between a pair of alarms, the correlation between them should be evaluated, with distance being an important factor. The distance can be classified into three categories, each measuring different aspects.

Suppose two alarms a_1, a_2 and their corresponding locations l_1, l_2 are given. The physical distance is the length of the shortest path between l_1 and l_2 in the network state, while the OMS distance is the number of OMS trails in this shortest path. Additionally, there is the OCh distance, which is a binary value that indicates whether there is a lightpath traversing both l_1 and l_2.

To evaluate the distance, ArangoDB is employed once again to manage the network state at the board level. A vertex is generated for each board, while an edge is established for every fiber. Furthermore, an edge is formed for each OMS trail within the graph. A visual representation of an 8-node network state is provided in Figure 3.21 for reference.

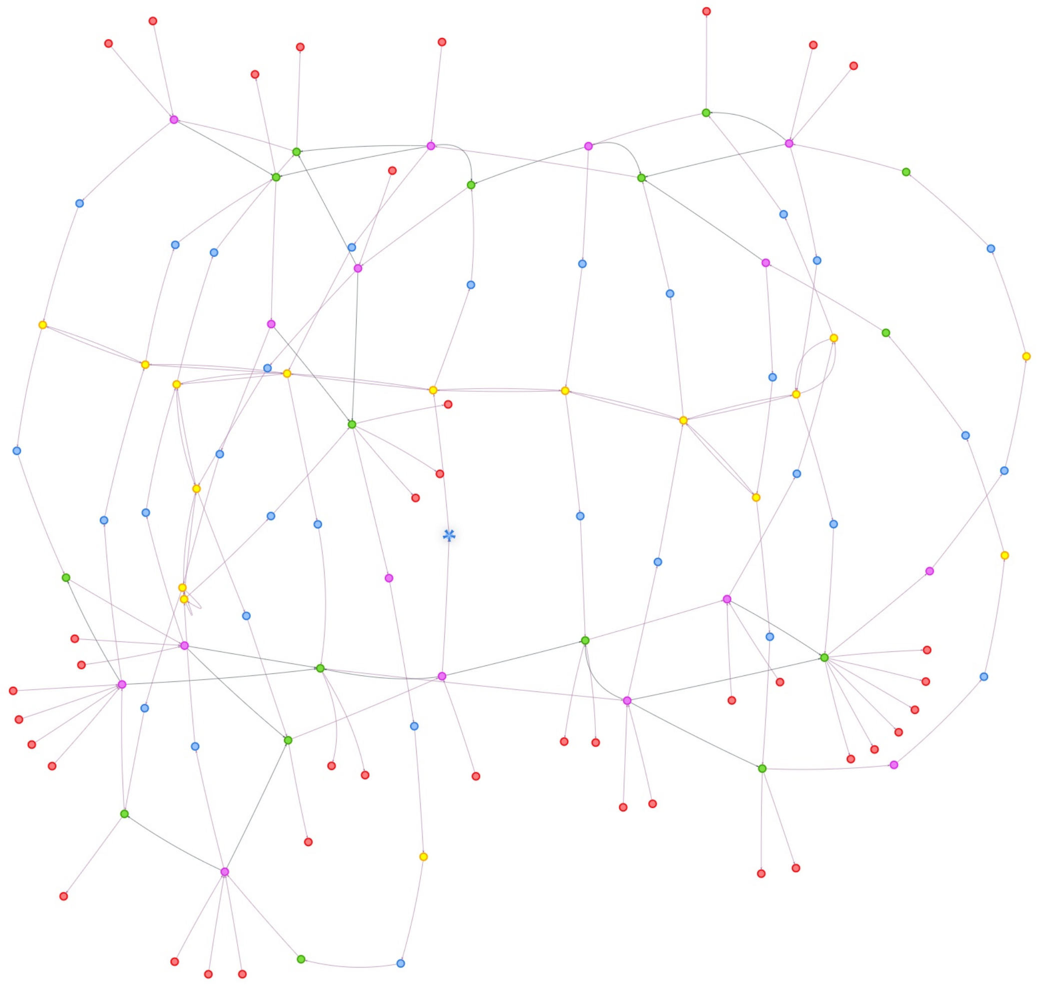


Figure 3.21: Example of a Network State in ArangoDB

To determine the physical and OMS distance, ArangoDB's built-in graph traversal query is applied to compute the shortest directional path between any two locations. Then, the length of this shortest path is calculated, representing the physical distance between the locations. If such path does not exist, then the physical distance is assigned a value of -1. Additionally, given the shortest directional path, the number of OMS trails traversed is counted, representing the OMS distance.

The calculation of the OCh distance does not utilize ArangoDB. Instead, for each pair of locations, every lightpath will be inspected to determine if both locations are traversed by a common lightpath.

Chapter 4

Case Studies

In this chapter, one static topology and one dynamic topology will be generated to evaluate the performance of the topology generator. Furthermore, to validate the functionality of the failure generator and alarm generator, three test cases will be introduced for static topology, encompassing single-failure, double-failure and noisy alarm events. Double-failure is the event when two failures happen at the same time, resulting in their alarms being intertwined in the timeline. Since the primary distinction between static topology and dynamic topology lies in the number of network states present, dynamic topology can be conceptualized as a composite of multiple static topologies. Hence, it suffices to illustrate the single-failure case for the dynamic topology.

4.1 Static Topology Scenario

The topology and traffic generated are displayed in Figure 4.1 and Table 4.2, with corresponding input parameters outlined in Table 4.1. Each lightpath in the figure is labeled with numbers indicating the nodes it traverses. For example, lightpath$_{2-1-0}$ indicates a lightpath starting from an OTU in ROADM2, passing through ROADM1, and terminating at an OTU in ROADM0.

It is important to note that a connection between ROADM0 and ROADM1 (R0 $\rightarrow$ R1) implies the existence of two fibers: ROADM0-FIU2 $\rightarrow$ ROADM1-FIU1 and ROADM1-FIU1 $\rightarrow$ ROADM0-FIU2. Therefore, if a lightpath traverses from ROADM1-FIU1 to ROADM0-FIU2, it visually appears to be in the opposite direction of the node connection depicted in the figure.

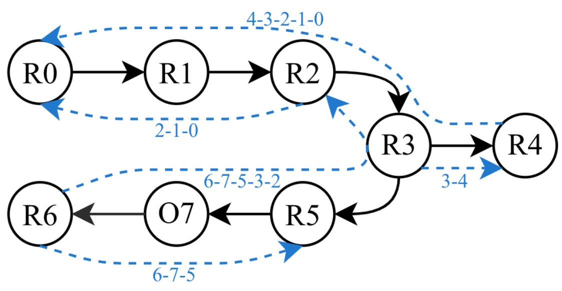


Figure 4.1: Network State in Static Topology Scenario

	Number of Nodes	Number of Lightpaths
Network State 1	8	5

Table 4.1: Parameters Input to Static Topology Generator

(OTU68)→ROADM4→ROADM3→ROADM2→ROADM1→ROADM0→(OTU95)
(OTU4)→ROADM6→OLA7→ROADM5→(OTU75)
(OTU28)→ROADM2→ROADM1→ROADM0→(OTU79)
(OTU26)→ROADM3→ROADM4→(OTU37)
(OTU16)→ROADM6→OLA7→ROADM5→ROADM3→ROADM2→(OTU35)

Table 4.2: Traffic in Static Topology Scenario

4.1.1 Single-failure Case

When a location type of "OA" is provided, the failure generator produces a failure as shown in Table 4.3, and the alarm generator generates a corresponding list of alarms, as displayed in Table 4.4, based on this failure. Furthermore, the alarm correlation tree can be constructed, as depicted in Figure 4.2.

Alarm type	Location	Time	ID	Failure ID	Is root	Is noisy
board faulty	ROADM0-OA1	06:00:00	0	0	false	false

Table 4.3: Failure for Single Failure Case in Static Topology Scenario

Alarm type	Location	Time	ID	Failure ID	Is root	Is noisy
board faulty	ROADM0-OA1	06:00:00	0	0	false	false
OTS_LOS_C	ROADM0-FIU2	06:00:10	182	0	true	false
OTS_A_P	ROADM0-OA1	06:00:07	183	0	true	false
OTS_BDI_A	ROADM0-FIU1	06:00:13	184	0	false	false
OMS_LOS_A	ROADM1-OM1	06:00:15	185	0	false	false
OMS_SSF_P	ROADM1-OD1	06:00:15	186	0	false	false
OTS_PMI	ROADM1-FIU1	06:00:16	187	0	false	false
OMS_A_P	ROADM0-OD1	06:00:16	188	0	false	false
OMS_SSF	ROADM0-OD1	06:00:15	189	0	false	false
OMS_SSF_E	ROADM2-OD1	06:00:24	190	0	false	false

Table 4.4: Alarms for Single Failure Case in Static Topology Scenario

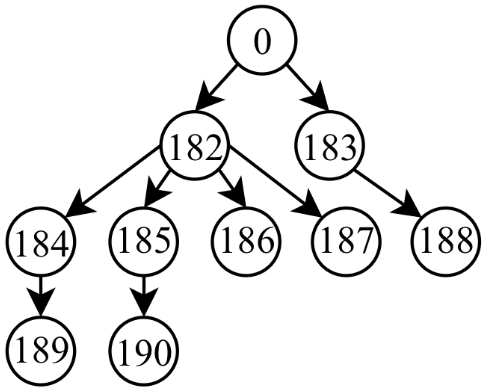


Figure 4.2: Alarm Correlation Tree for Single Failure Case in Static Topology Scenario

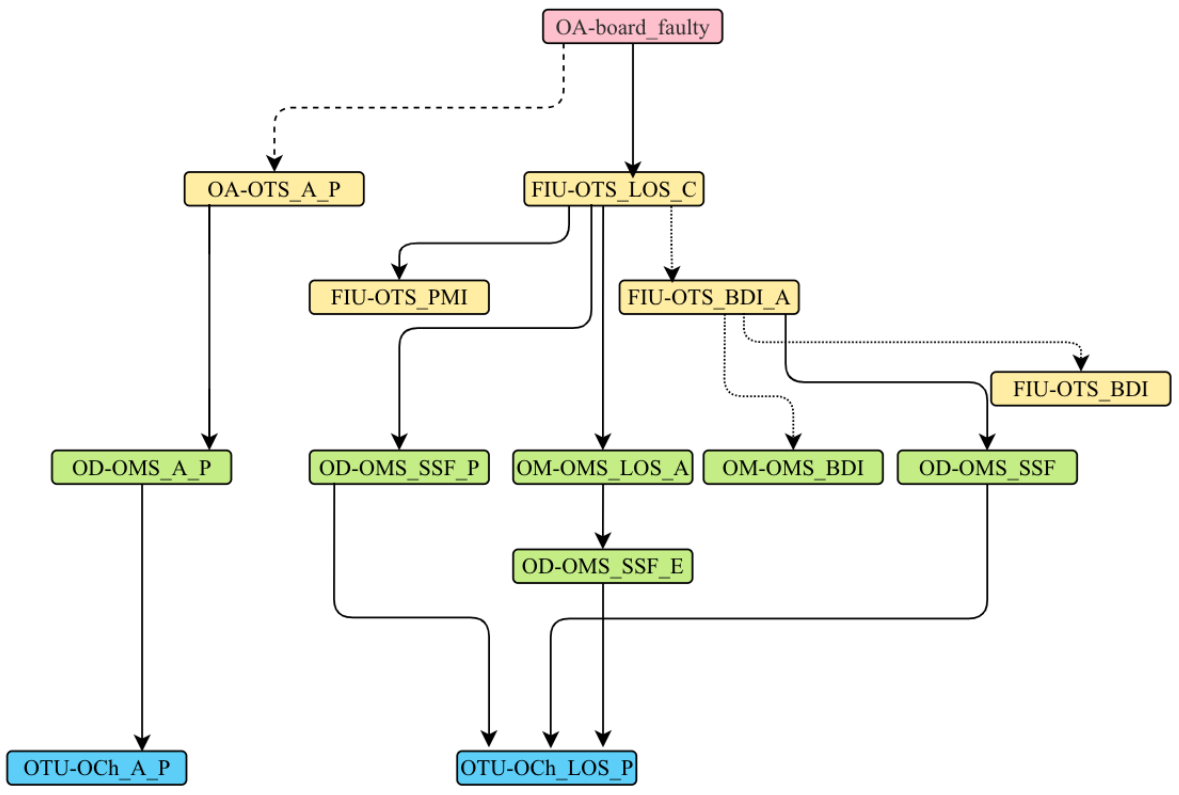


Figure 4.3: Rule Events for OA Board Faulty

Given the subset of the rule database indicating a failure at OA, we can validate the accuracy of the alarm generator. A faulty board located on ROADM0-OA1 will initiate a root alarm, OTS_LOS_C, which will propagate downstream to ROADM0-FIU2. Furthermore, another root alarm OTS_A_P is locally reported on ROADM0-OA1. Subsequently, OTS_A_P triggers OMS_A_P. Since there is no lightpath traversing both ROADM0-OA1 and ROADM0-OD1, the alarm OCh_A_P will not be triggered.

Similarly, OTS_LOS_C initiates a series of alarms including OTS_BDI_A, OMS_LOS_A, OMS_SSF_P and OTS_PMI, where OTS_BDI_A and OMS_LOS_A further trigger OMS_SSF and OMS_SSF_E respectively. Notably, apart from OCh_LOS_P, neither OMS_BDI nor OTS_BDI alarms are activated. This is due to the absence of OM and FIU boards upstream of ROADM0-FIU1.

4.1.2 Double-failure Case

Given two location types "OTU" and "FIU_FIU", along with the double-failure time format, the failure generator generates two failures as depicted in Table 4.5, where both failures occur at the same time.

Alarm type	Location	Time	ID	Failure ID	Is root	Is noisy
board faulty	ROADM6-OTU16	06:00:00	79	79	false	false
fiber cut	ROADM1-FIU1_ ROADM0-FIU2	06:00:00	102	102	false	false

Table 4.5: Failure for Double Failure Case in Static Topology Scenario

Alarm type	Location	Time	ID	Failure ID	Is root	Is noisy
board faulty	ROADM6-OTU16	06:00:00	79	79	false	false
fiber cut	ROADM1-FIU1_ ROADM0-FIU2	06:00:00	102	102	false	false
OCh_LOS_P	ROADM2-OTU35	06:00:05	182	79	true	false
OCh_A_P	ROADM6-OTU16	06:00:07	183	79	true	false
OTS_LOS_O	ROADM0-FIU2	06:00:07	184	102	true	false
OTS_BDI_A	ROADM1-FIU1	06:00:09	185	102	false	false
OMS_BDI_O	ROADM1-OM2	06:00:08	186	102	false	false
OMS_SSF_O	ROADM0-OD2	06:00:11	187	102	false	false
OTS_BDI	ROADM1-FIU2	06:00:16	188	102	false	false
OMS_BDI	ROADM1-OM2	06:00:17	189	102	false	false
OMS_SSF	ROADM0-OD2	06:00:16	190	102	false	false
OCh_LOS_P	ROADM0-OTU95	06:00:14	191	102	false	false
OCh_LOS_P	ROADM0-OTU79	06:00:16	192	102	false	false
OCh_LOS_P	ROADM0-OTU95	06:00:22	193	102	false	false
OCh_LOS_P	ROADM0-OTU79	06:00:19	194	102	false	false

Table 4.6: Alarms for Double Failure Case in Static Topology Scenario

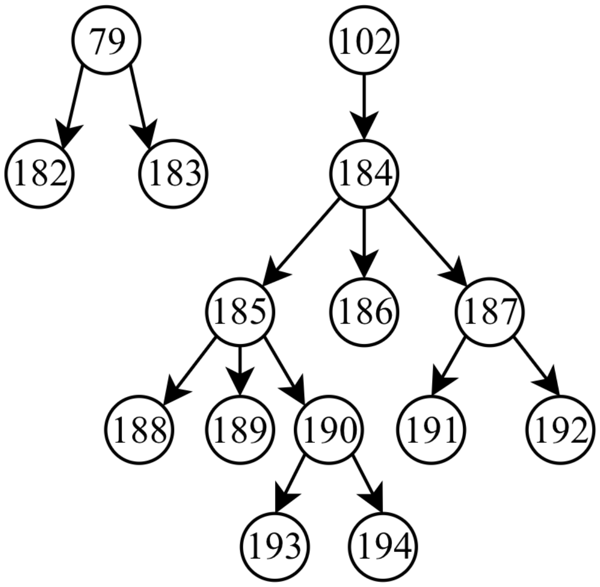


Figure 4.4: Alarm Correlation Tree for Double Failure Case in Static Topology Scenario

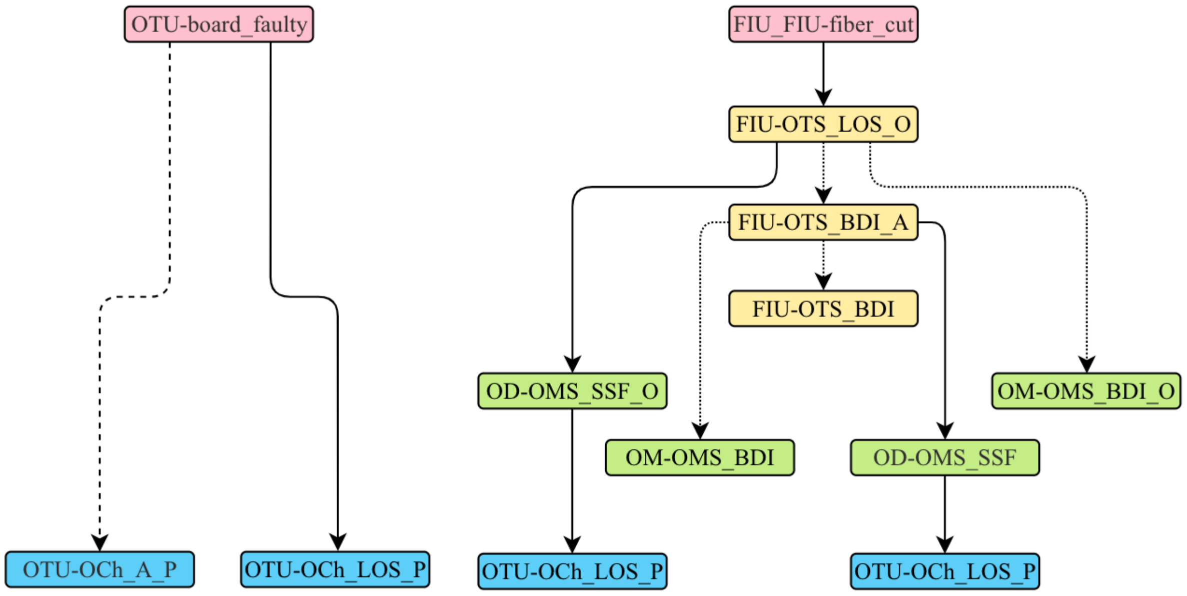


Figure 4.5: Rule Events for OTU Board Faulty and FIU_FIU Fiber Cut

The alarms and the alarm correlation trees resulting from the failures are presented in Table 4.6 and Figure 4.4. In the case of an OTU board faulty, two OCh alarms are generated: one is locally reported, and the other is transmitted downstream. For the FIU_FIU fiber cut, eleven alarms are generated, encompassing all possible rule events that could be triggered.

Additionally, it can be observed that the alarms generated by these two failures overlap both in timeline and alarm type, thereby increasing the complexity of failure localization.

4.1.3 Noisy Alarm Case

To assess the resilience of failure localization methods, noisy alarms are introduced to the alarm sets. In addition to the essential parameters for failure generation, the alarm generator requires a "noisy ratio" to determine the number of noisy alarms to add. This ratio denotes the proportion of noisy alarms to real alarms and typically ranges between 0.1 and 1.

To produce the noisy alarms, the initial step involves counting the number of real alarms to determine the quantity of noisy alarms. Subsequently, failures are generated based on a randomly selected list of locations within the network state, followed by the generation of alarms stemming from these failures. Finally, noisy alarms are chosen from this alarm set, with the desired count, where the *Is Noisy* field is designated as 1.

Alarm type	Location	Time	ID	Failure ID	Is root	Is noisy
board faulty	ROADM0-OA1	06:00:00	0	0	false	false
OTS_LOS_C	ROADM0-FIU2	06:00:02	182	0	true	false
OTS_A_P	ROADM0-OA1	06:00:09	183	0	true	false
OTS_BDI_A	ROADM0-FIU1	06:00:05	184	0	false	false
OMS_LOS_A	ROADM1-OM1	06:00:05	185	0	false	false
OMS_SSF_P	ROADM1-OD1	06:00:07	186	0	false	false
OTS_PMI	ROADM1-FIU1	06:00:08	187	0	false	false
OMS_A_P	ROADM0-OD1	06:00:11	188	0	false	false
OMS_SSF	ROADM0-OD1	06:00:06	189	0	false	false
OMS_SSF_E	ROADM2-OD1	06:00:06	190	0	false	false
OMS_SSF	ROADM4-OD1	06:00:08	191	0	false	true
OMS_LOS_A	ROADM5-OM1	06:00:08	192	0	false	true
OMS_BDI	ROADM3-OM1	06:00:06	193	0	false	true
OTS_LOS_C	ROADM5-FIU1	06:00:03	194	0	false	true

Table 4.7: Alarms for Noisy Alarm Case in Static Topology Scenario

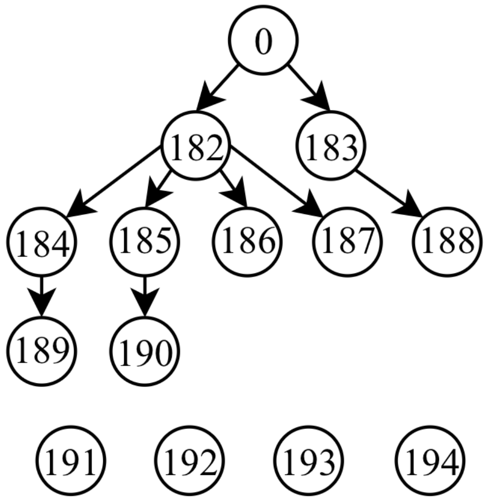
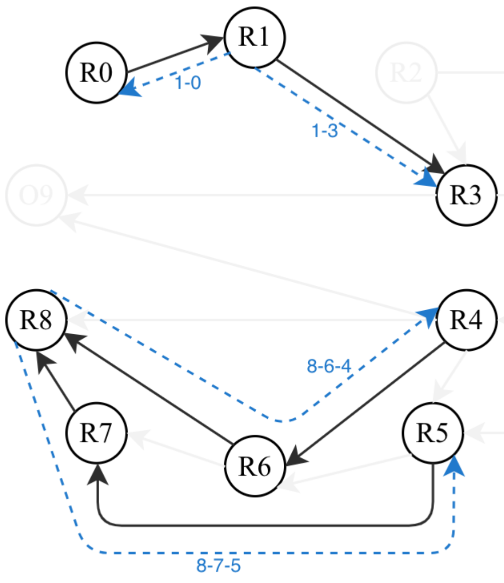


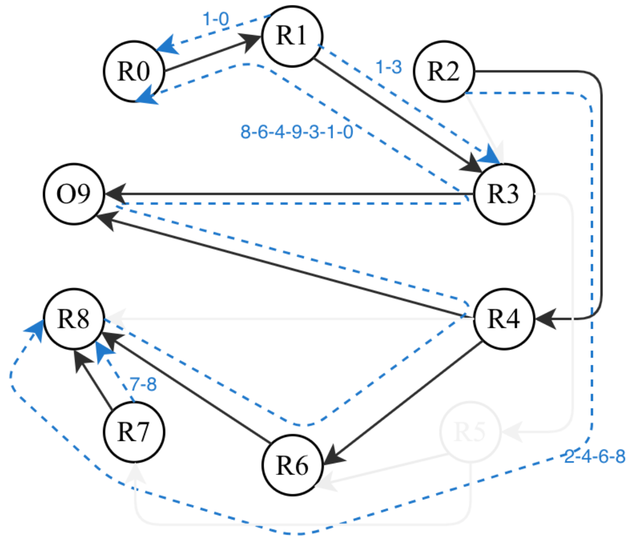
Figure 4.6: Alarm Correlation Tree for Noisy Alarm Case in Static Topology Scenario

Utilizing the failure from Table 4.3 and a noisy ratio of 0.5, the alarm generator produces a list of alarms detailed in Table 4.7, accompanied by the correlation tree depicted in Figure 4.6. Notably, compared to Table 4.4, the alarm set now includes four additional noisy alarms. Additionally, the alarm correlation tree exhibits the addition of four dangling vertices.

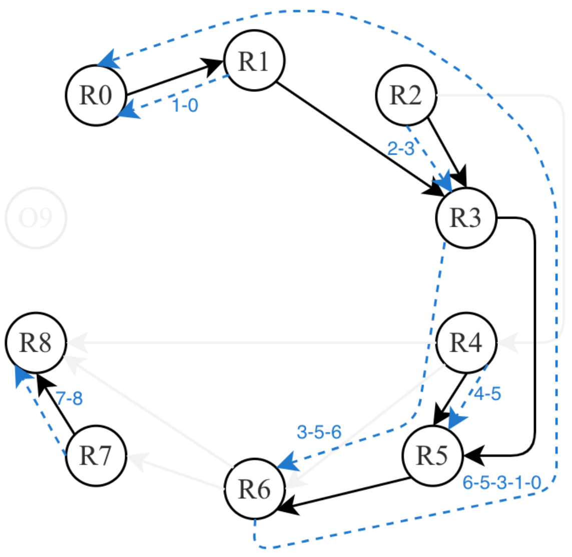
4.2 Dynamic Topology Scenario



(a) Network State 1



(b) Network State 2



(c) Network State 3

Figure 4.7: Network States in Dynamic Topology Scenario

	Number of Nodes	Number of Lightpaths
Network State 1	10	4
Network State 2		5
Network State 3		6

Table 4.8: Parameters Input to Dynamic Topology Generator

Given the inputs outlined in Table 4.8, the topology generator returns a topology with three network states, depicted in Figure 4.7, alongside the corresponding traffics delineated in Tables 4.9, 4.10, and 4.11. It is noticeable that each network state utilizes a subset of nodes and connections within the entire network. Furthermore, the traffics exhibit duplicate lightpaths, simulating scenarios where lightpaths are added and removed over time.

(OTU44)→ROADM1→ROADM0→(OTU87)
(OTU72)→ROADM8→ROADM7→ROADM5→(OTU3)
(OTU2)→ROADM1→ROADM3→(OTU17)
(OTU104)→ROADM8→ROADM6→ROADM4→(OTU83)

Table 4.9: Traffic in Network State 1

(OTU44)→ROADM1→ROADM0→(OTU87)
(OTU8)→ROADM8→ROADM6→ROADM4→OLA9→ROADM3→ROADM1→ROADM0→(OTU3)
(OTU18)→ROADM2→ROADM4→ROADM6→ROADM8→(OTU5)
(OTU106)→ROADM7→ROADM8→(OTU41)
(OTU2)→ROADM1→ROADM3→(OTU17)

Table 4.10: Traffic in Network State 2

(OTU106)→ROADM7→ROADM8→(OTU41)
(OTU16)→ROADM6→ROADM5→ROADM3→ROADM1→ROADM0→(OTU43)
(OTU90)→ROADM2→ROADM3→(OTU77)
(OTU44)→ROADM1→ROADM0→(OTU87)
(OTU110)→ROADM4→ROADM5→(OTU45)
(OTU2)→ROADM3→ROADM5→ROADM6→(OTU1)

Table 4.11: Traffic in Network State 3

4.2.1 Single-failure Case

Alarm type	Location	Time	ID	Failure ID	Is root	Is noisy
board faulty	ROADM3-FIU2	06:00:00	61	61	false	false

Table 4.12: Failure for Single Failure Case in Network State 1

Alarm type	Location	Time	ID	Failure ID	Is root	Is noisy
board faulty	ROADM3-FIU2	06:00:00	29	29	false	false

Table 4.13: Failure for Single Failure Case in Network State 2

Alarm type	Location	Time	ID	Failure ID	Is root	Is noisy
board faulty	ROADM3-FIU2	06:00:00	50	50	false	false

Table 4.14: Failure for Single Failure Case in Network State 3

To investigate the influence of network state and traffic on alarm propagation, the failure event with the same location is applied to all three network states. The outcomes are presented in Tables 4.15, 4.16, 4.17 and Figures 4.8, 4.9, 4.10.

Alarm type	Location	Time	ID	Failure ID	Is root	Is noisy
board faulty	ROADM3-FIU2	06:00:00	61	61	false	false
OTS_A_P	ROADM3-FIU2	06:00:07	188	61	true	false

Table 4.15: Alarms for Single Failure Case in Network State 1

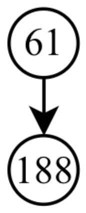


Figure 4.8: Alarm Correlation Tree for Single Failure Case in Network State 1

Alarm type	Location	Time	ID	Failure ID	Is root	Is noisy
board faulty	ROADM3-FIU2	06:00:00	29	29	false	false
OTS_LOS_C	OLA9-FIU1	06:00:02	204	29	true	false
OTS_A_P	ROADM3-FIU2	06:00:10	205	29	true	false
OTS_BDI_A	ROADM3-FIU2	06:00:10	206	29	false	false
OMS_LOS_A	ROADM4-OM1	06:00:08	207	29	false	false
OMS_SSF_P	ROADM4-OD1	06:00:08	208	29	false	false
OTS_PMI	OLA9-FIU2	06:00:06	209	29	false	false
OTS_BDI	ROADM3-FIU1	06:00:13	210	29	false	false
OMS_BDI	ROADM3-OM1	06:00:13	211	29	false	false
OMS_SSF	ROADM4-OD1	06:00:19	212	29	false	false
OMS_SSF_E	ROADM6-OD1	06:00:14	213	29	false	false

Table 4.16: Alarms for Single Failure Case in Network State 2

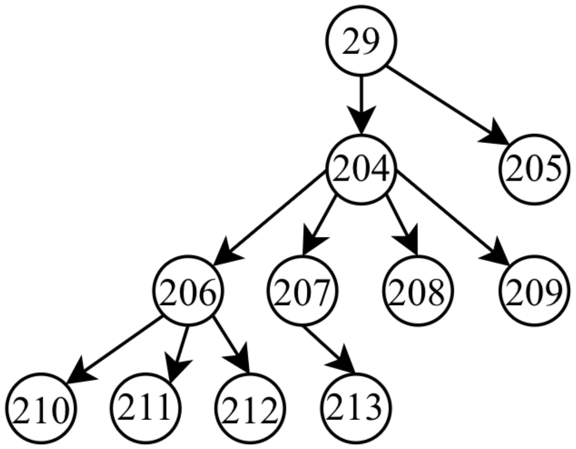


Figure 4.9: Alarm Correlation Tree for Single Failure Case in Network State 2

Alarm type	Location	Time	ID	Failure ID	Is root	Is noisy
board faulty	ROADM3-FIU2	06:00:00	50	50	false	false
OTS_LOS_C	ROADM5-FIU1	06:00:06	218	50	true	false
OTS_A_P	ROADM3-FIU2	06:00:03	219	50	true	false
OTS_BDI_A	ROADM3-FIU2	06:00:13	220	50	false	false
OTS_BDI_A	ROADM4-FIU2	06:00:15	221	50	false	false
OMS_LOS_A	ROADM5-OM1	06:00:09	222	50	false	false
OMS_SSF_P	ROADM5-OD1	06:00:10	223	50	false	false
OTS_PMI	ROADM5-FIU2	06:00:13	224	50	false	false
OTS_BDI	ROADM3-FIU1	06:00:22	225	50	false	false
OMS_BDI	ROADM3-OM1	06:00:20	226	50	false	false
OMS_SSF	ROADM5-OD1	06:00:21	227	50	false	false
OTS_BDI	ROADM4-FIU1	06:00:17	228	50	false	false
OMS_BDI	ROADM4-OM1	06:00:20	229	50	false	false
OMS_SSF	ROADM5-OD1	06:00:24	230	50	false	false
OMS_SSF_E	ROADM6-OD1	06:00:11	231	50	false	false
OCh_LOS_P	ROADM6-OTU1	06:00:20	232	50	false	false
OCh_LOS_P	ROADM6-OTU1	06:00:23	233	50	false	false
OCh_LOS_P	ROADM6-OTU1	06:00:30	234	50	false	false
OCh_LOS_P	ROADM6-OTU1	06:00:21	235	50	false	false

Table 4.17: Alarms for Single Failure Case in Network State 3

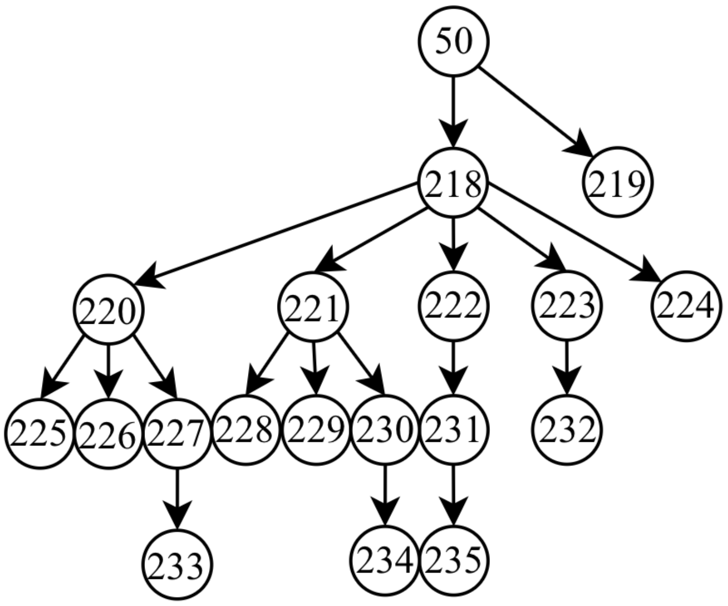


Figure 4.10: Alarm Correlation Tree for Single Failure Case in Network State 3

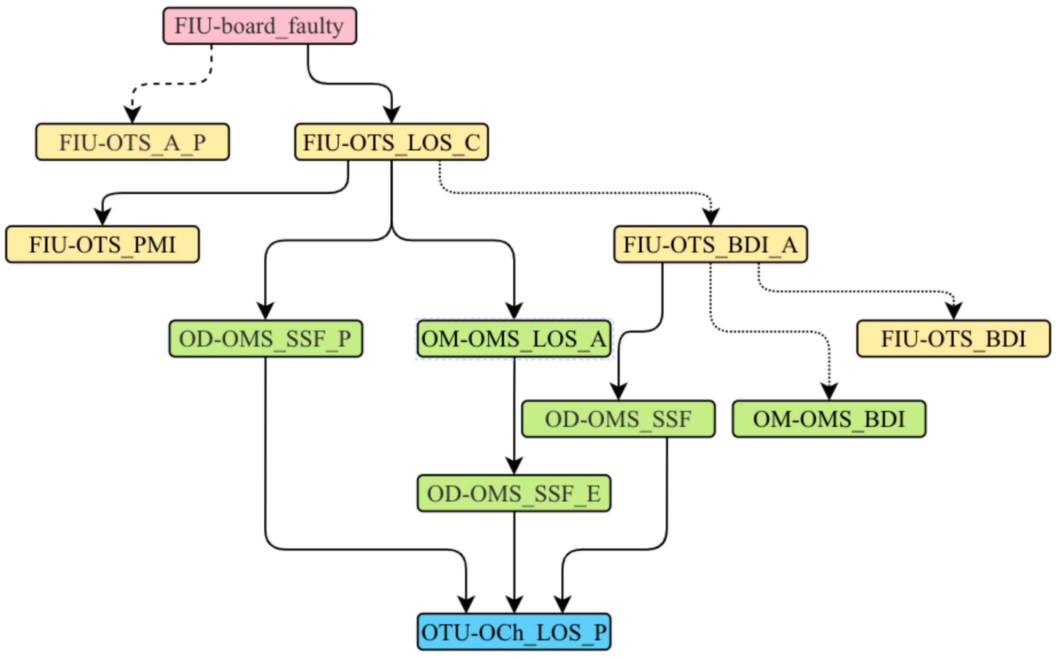


Figure 4.11: Rule Events for FIU Board Faulty

Chapter 5

Conclusion

This thesis presents the design and implementation of a simulator aimed at replicating alarm propagation behavior across diverse network topologies. Building upon the framework outlined in the previous iteration referenced in [5], the updated version introduces a topology generator capable of accommodating both static and dynamic network configurations. Additionally, enhancements encompass the integration of board-level network states, random propagation time, alarm ID, and the introduction of noisy alarms. The simulator is structured around the following key components:

- A rule database storing propagation rules for each alarm/failure type.
- A topology generator capable of producing static or dynamic topologies with various settings.
- A failure generator that generates failures based on a list of locations or location types.
- An alarm generator responsible for generating alarms from failures.

After outlining the design and functionalities of the simulator, it is imperative to address its limitations:

- While the simulator effectively propagates alarms across OTS, OMS, and OCh layers, it currently overlooks the digital layers. Future enhancements should encompass the digital layers within the simulator to provide a more comprehensive representation of OTN.

- Manual editing of the rule database is currently required, which is time-consuming and prone to errors. Additionally, the rule database lacks consideration for the many-to-one case, where a rule event may be triggered by multiple rule events simultaneously. To address these shortcomings, future updates of the simulator could incorporate a rule generator capable of efficiently generating more complex rules.

- The current simulation framework lacks support for a sufficient range of node types and board types, limiting its ability to accurately simulate the complexities inherent in OTN. Expansion of the simulator to include a broader node and board types would enable more realistic modeling of OTN environments.

Furthermore, there is potential for further improvement by extending the simulator's support beyond OTN to encompass general fiber optical communication systems.

References

[1] "Ethernet Interfaces User Guide for Routing Devices," 2023. [Online]. Available: https://www.juniper.net/documentation/us/en/software/junos/interfaces-ethernet/topics/topic-map/ethernet-otn-options-overview.html

[2] "INFOGRAPHIC: Optical Transport Network." [Online]. Available: https://www.ciena.com/insights/infographics/OTN-poster.html?utm_source=Blog&utm_medium=Social

[3] R. Ramaswami, K. N. Sivarajan, and G. H. Sasaki, *Optical Networks: A Practical Perspective*. Morgan Kaufmann, 2009. [Online]. Available: https://www.sciencedirect.com/book/9781558606555/optical-networks#book-info

[4] "G.709: Interfaces for the optical transport network," International Telecommunication Union, ITU-T Recommendation, 2020. [Online]. Available: https://www.itu.int/rec/T-REC-G.709-202006-I/en2020

[5] Z. Li, P.-H. Ho, Y. Jiao, B. Li, and Y. You, "Design of an OTN-based Failure/Alarm Propagation Simulator," in *2022 International Conference on Networking and Network Applications*, 2022, pp. 1–5.

[6] A. Schibert, "White Paper: G.709 – The Optical Transport Network (OTN)," VIAVI Solutions, Tech. Rep., 2021. [Online]. Available: https://www.viavisolutions.com/en-us/literature/g709-optical-transport-network-otn-white-papers-books-en.pdf

[7] "OTN VS SONET/SDH," 2023. [Online]. Available: https://community.fs.com/article/otn-vs-sonetsdh.html

[8] Y. Jiao, P.-H. Ho, X. Lu, K. Liang, Y. You, J. Tapolcai, B. Li, and L. Peng, "On real-time failure localization via instance correlation in optical transport networks," in *2023 IFIP Networking Conference (IFIP Networking)*, 2023, pp. 1–9.