GMPLS/Lightwave Agile Switching Simulator

Oliver Borchert

GLASS January Workshop
January 6th to January 10th 2003

Software – www.antd.nist.gov/glass
Questions – glass@antd.nist.gov

NIST Participants
- NIST ANTD – David Su / Division Chief
  - High Speed Network Technologies Group
    - Optical Framework and Graphical Tools
      - Nada Golmie – Group Manager
      - Oliver Borchert – Project lead, Stephan Klink
      - Richard Rouil
    - Internetworking Group
      - MPLS / Diff Serv / TE
      - Doug Montgomery – Group Manager
      - Chul Kim
      - Young-tak Kim, Kwang-il Lee, Eun-hyuk Lim
Agenda (1)

- **Monday**
  - Introduction
  - SSFNet Overview
  - Optical Components
  - The TSC in general
- **Lunch**
  - Events in GLASS
  - TSC Event Panel
  - ONIC and failure detection / propagation
  - The Optical Frame Header

- **Tuesday**
  - Optical Protocols
    - The OXCSwitch
    - Signaling Protocol Example
  - Lunch
  - Algorithms
  - The Optical Path Structure
  - Utilities
  - The TSC Configuration

Agenda (2)

- **Wednesday**
  - GMPLS
  - Lunch
    - Splitting into 2 groups for the advanced sessions.
- **Thursday**
  - Advanced session
  - Lunch
  - Advanced session

- **Friday**
  - Advanced session
  - Lunch
  - Ending discussion

**Advanced Session:**
- Group1:
  - Integration of Algorithms
  - interfacing with TSC
- Group2L
  - Data Structure of new GMPLS and DML schema.
  - Scripting scenarios
**Introduction**

- This workshop is scheduled on request of the IOWA state University.
- The workshop contains two “advanced sessions” in parallel and one joint
  - Joint:
    - Overview over the GLASS framework.
  - Parallel:
    - Integration of Algorithms into GLASS and the TSC.
    - Modeling and design of GMPLS scenarios.

**First Intension**

- Allow the development and evaluation of R&WA algorithms.
- Create a tool to facilitate:
  - the easy planning of a WDM network.
  - the analysis for WDM protocol performance.
- Test a network constellation before building it in reality.
Our step to SSF (1)

- NIST already had a “simulator” called MERLiN
- Simulation had to be implemented in the protocol itself.
- No discrete simulation framework users could use.

Our step to SSF (2)

- We checked simulator such as:
  - NS2
  - JavaSim (OHIO State University)
  - SSF/SSFNet
- We had 3 teams to test the simulation frameworks.
- We tested the tools for an approximately time of 2-3 Months.
- Final decision was SSFNet – It was already used at NIST
What is GLASS?

- GLASS is an extension of the Scalable Simulation Framework Network (SSFNet).
- It contains:
  - An optical framework
  - MPLS framework
- Allows the modeling and performance evaluation, restoration, and developing of signaling protocols for optical / GMPLS networks.

GOALS

- Enhance the NIST optical network modeling tool, MERLiN, to include discrete event simulation.
- Provide models to evaluate architectures and protocols for protection and restoration, routing, signaling and management of GMPLS / Optical networks.
- Support multi-level / multi-protocol schemes for traffic engineering, QoS, protection and restoration.
GLASS Project Statistics

- 7 java libraries.
- 86 packages.
- 717 classes.
- 508 java source files.
- 129,673 lines of code.
- 18,236 lines of documentation.
- 1 Command line Simulator.
- 2 Graphical Simulator tools.
  - TSC and Browser

GLASS Components (1)

Optical Components:
- Optical Network Interface
  - WDM interfaces
  - λ / fiber level attributes, behavior models
  - Failure notification
- Optical Cross Connects
  - Add-drop ports
  - Switching - λ, port
  - λ conversion
  - OXC edge router
- Algorithms (RWA, protection)
  - Best-Fit, SPF, SPF-SRLG
  - Link protection, restoration
  - More are produced by GWU

MPLS Components:
- MPLS Core Switching
  - LSRs, LERs
  - E-LSPs, L-LSPs
  - Multi Field Classifier
  - Hierarchical LSPs
  - LSP metering
- Signaling / Routing
  - CRLDP (TE,GMPLS), OAM
  - s-OSPF-(TE,GMPLS)
- Traffic Engineering
  - TE Agent: OSPF-TE, CRLDP, event notifications
GLASS Components (2)

- **Diff Serv Components:**
  - Metering: TBM, TSW, SRTCM, TRTCM
  - Queue Management: Drop tail, RED, RIO
  - Scheduling: Priority, WRR, WFQ, Hierarchical combos

- **SSF Environment:**
  - Topology explorer
  - Failure scripting
  - Interactive simulation
  - Protocol Animator
  - Network designer

More than just new Components

- Support of shared risk strategies for failure recovery. (SRG & SRLG)
- Facilitate customized granularity of failure detection. (Failure Recovery Module - FRM)
- Support monitoring of incoming traffic.
- Scripting of events such as failures or recoveries.
- Providing an internal data structure for algorithm and protocol writers.
- Providing a generic algorithm interface.
SSFNet Overview

SSFNet:
- A collection of Java SSF-based components.
- Modeling and simulation of Internet protocols and networks at and above the IP packet level of detail.
- Each component configures itself through a configuration database.
- Use Data Modeling Language (DML) to script network topology and simulation scenarios.

The Package SSF.OS contains components for modeling the basis framework.
The Package SSF.Net contains components for modeling network connectivity as well as components for creating node and link configurations.

The Package SSF.Net

- Net
- Host and Router
- NIC
- Link
The Package SSF.OS

- ProtocolGraph
- ProtocolSession
- ProtocolMessage
- PacketEvent

The ProtocolMessage

- The object ProtocolMessage represents the message exchanged between the different ProtocolSessions (protocol layers) in a node (ProtocolGraph).
- The basic idea of a ProtocolMessage is the same as the real world: a Protocol Message has a header and a payload. The payload itself is also a ProtocolMessage.
The ProtocolMessage (2)

- The class ProtocolMessage provides the necessary methods for manipulation (add/remove payload).

DML - The Domain/Data Modeling Language

- The DML syntax specifies a hierarchy of lists of attributes (key-value pairs).
- It is used for:
  - The model description
  - Model instantiation (first phase of simulation)
  - Runtime inspection

2 important interfaces:
- Configuration:
  - used by the classes that store and provide access to DML configurations in memory (the database objects).
- Configurable:
  - used by any class that can self-configure itself using the attributes obtained from the matching Configuration objects (protocols, interface...).
Scripting a Network in DML

Net

host
1 id
1 interface

router
2 id
2 interface

link
1(1) attach
2(1) attach

Net

host
1 id
1 interface

router
2 id
2 interface

link
1(1) attach
2(1) attach

Scripting Protocols in DML

protocal graph

router graph

ProtocolSession [name tcp use SSF.OS.TCP.tcpSessionMaster]
ProtocolSession [name ip use SSF.OS.IP]
Simulation monitoring (1)

Measurement of...
- end-2-end application data.
- internal state of protocol sessions.
- the queue monitor.
- routing and wavelength assignment update.
- ...

Characteristics:
- Flexibility: Automatic configuration of the monitors in multiple nodes: use of dml file to place the monitors and to configure them.
- Fast output: use byte streams of records and source multiplexing.
- Fast record retrieval: demultiplex records.

Simulation monitoring (2)

- The user configures via the DML file where the monitors are installed (protocols and queues).
- During the simulation, monitors write a byte stream into one or more files.
- After the simulation, the user uses a player to retrieve the collected information and formats them.

The user configures via the DML file where the monitors are installed (protocols and queues).
- During the simulation, monitors write a byte stream into one or more files.
- After the simulation, the user uses a player to retrieve the collected information and formats them.

Monitoring Input Traffic

- Provides the capability to use the SSF monitor framework for incoming traffic.
- Allows to monitor loss of packages in the link layer.
- Facilitates the collection of raw data of message delays in the link layer.

Multi timeline in SSFNet

- Each node can be in its own timeline.
- Direct access can cause a timeline conflict.
- Data exchange only by sending messages.
Timeline problematic (1)

Direct access by method call!

Is node down?

Receive node up!

Timeline 1

Timeline 2

Timeline problematic (1)

Timeline Conflict!

Received node 2 is up because of different timelines!

Timeline 1

Timeline 2
Timeline problematic (2)

Timeline solution (1)

Timeline Conflict!

Link failure at time t2

Failure?

Yes!

Timeline 1

Timeline 2

Timeline 1

Timeline 2

Link failure at t2

Failure?

Yes!

No!

Process p
Timeline solution (1)

- Link failure at time $t_2$
- Failure? Yes!

Timeline 1
- $t_0$, $t_1$, $t_2$

Timeline 2
- $t_0$, $t_1$, $t_2$

Optical Components
The Optical Components

- Routers and components:
  - Optical Cross Connect (OXC)
  - Edge Router (OXCEdgeRouter / LSR)
  - Optical Network Interface (ONIC)
  - Optical Switch (OXCSwitch)

- Link:
  - Optical Link – Optical Link Layer
  - Fiber
  - Lambda

- Optical Connection:
  - Optical Channel
  - Optical Channel Segment

The Optical Components class hierarchy

Diagram showing the class hierarchy of optical components, including SSFNet and GLASS, with classes such as Router, ExtRouter, LSR, OXCEdgeRouter, OXC, and pure optical links.
**Label Switched Router (LSR)**

- The Protocol must be able to communicate with the NIC and ONIC.
- The switching must be done in the protocol layer.

**Optical Links**

- The basic medium to transport a message through an optical network is a fiber.
- Can we use the link of SSF as representation of a fiber?
The Optical Link

- Consists of:
  - A bundle of fibers
  - Each fiber contains a list of lambdas
Scripting an Optical Link

```
OpticalLink {
  id 3
  attach 3(0) attach 4(1)
  distance 1
  delay 0.00005
}
```

No Fibers specified:
- automat-generation of two unidirectional Fibers
- Each fiber has ten Lambdas with 2.5 Gig.
  - one control (red)
  - 9 data (black)

The Optical Link Layer

- Is an abstract helper to create channels for the “entity” communication.
Scripting a Fiber (1)

OpticalLink [
id 3
attach 3(0) attach 4(1)
distance 1
delay 0.00005
]

Fiber [
    id 0
    host2Nhi 4
    host2PortID 0
    host1PortID 5
    noLambdas 4
    bidirectional true
]

Scripting a Fiber (2)

OpticalLink [
    id 3
    attach 3(0) attach 4(1)
distance 1
delay 0.00005
]

Fiber [
    id 0
    host2Nhi 4
    host2PortID 0
    host1PortID 5
    noLambdas 4
    bidirectional false
]
Scripting a Lambda

Fiber
- id 0
  - host2 Nhi
  - host1 PortId
  - host2 PortId

Lambda
- id 0
  - wavelength 1500
  - bandwidth 2.5
  - control false
  - receiver host2

...
The Topology and Simulation Creator (TSC) is a graphical interface that provides:

- Creation and configuration of topology
  - Network
  - Hosts (protocols, interfaces)
  - Links
  - Algorithms
  - Connections ...
- Runtime simulation
  - Control of simulation (speed, breakpoints)
  - Protocols and messages debugging
  - Visualization of connections and packets.
The Framework provides a capability to manage topology events. These events are part of the simulation. They are configured in the DML. Events are located in the timeline of the affected component.
Types of events

- At this time we only have failure and recovery events.
- Script Event.
  - Node Event class DefaultNodeEvent
  - Link Event class DefaultLinkEvent
  - ONIC Event class DefaultONICEvent
  - Fiber Event class DefaultFiberEvent
  - Lambda Event class DefaultLambdaEvent
- These events manipulate the attribute "failure" in the components.

Example of Event injection

- Event Manager
  - Read event.
  - Install link event.
- Create timers for time 0.5 sec
- event [type link time 0.5 link Nhi 3 failure true]
Example of Event injection

Event Manager
- Read event.
- Install link event.

```
event [
    type link
    time 0.5
    linkId 3
    failure true
]
```

Demo TSC - Event Panel
The Network Interface Card

- **Problem:**
  - The Network Interface Card (NIC) is connected to
    - IP on the upper side
    - SSF link infrastructure on the lower side.

- **Solution:**
  - Rewrite the NIC
  - Subclass the NIC and create an Optical Network Interface Card (ONIC)

---

The ONIC class hierarchy

```
       NIC
        ▲
        ▲
      _NIC
        ▲
      ONIC
```

Used as NIC in SSFNet of GLASS
The ONIC

- The ONIC connects node to an optical link.
- The optical link is similar to multiple links. => Each lambda in a fiber similar to an independent link.
- Parallel message processing.
The ONIC Interface

- OpticalFrameHeader
- fiberID, lambdaID
- Payload (MESSAGE)
- push (message, session)
- ONIC

More (2)

**Hardware Failure Detection**

- Failure detection not only in the optical domain.
- We want to be able to detect failures in the link as well as in the optical link.
- Using the event scripting capability.
The Failure Module

- An interface to allow customized modules in the _NIC
- Detects the failure and produces the Event - Message
- Passes the message back to the _NIC

Event Message

- Used for Inter-Component-Communication (ICC) → Failure notification
- 3 Basic Message types:
  - Alarm Message
  - Error Message
  - Information Message
- 2 Specialized Messages:
  - Failure Message (Error Message Type)
  - Recover Message (Information Message Type)
NIC
The Failure Detection

Failure Handler

Register for Notification

Failure Notification via ICC

_Failure Event_

Failure Module
Process Failure
Process Recover

Notification via ICC
Failure Module

Register for Notification
Failure Handler

_Failure Event_