International Journal of Innovative Research in Advanced Engineering (IJIRAE) ISSN: 2349-2163
Volume 1 Issue 8 (September 2014) www.ijirae.com

Resource Description Schema (MLRDS) and automatic path finding mechanism for Survivable Myltilayer Networks

Prof.H.B.Walikar,
Dept.of Computer Science Karnataka University Dharwad,

Ramesh.K,
Dept.of Computer Science Karnataka State Women's University Bijapur.

Abstract -- Our main contribution is to describe automated exchange of topology and state information between Optical hybrid networks consists of a routed IP part and circuit switched optical part (Light paths).This paper introduces an adaptive Multilayer Layer Resource Description Schema (MLRDS)[3] vocabulary designed to describe optical networks. MLRDS describes the schema that allows hybrid domains to exchange network state information which aids an application to query and find an alternate edge and path for survivable networks. We also have given some popular topology templates which can be reused to generate survivable multilayer networks [1] using this schema.

Key words – Survivable Multilayer Network, NDL, MLRDS, Lightpath, topology

I. Introduction

In recent years the design of research networks have been moving to multilayered hybrid network models[1].these networks consist of a routed IP part and circuit switched optical part (Light paths).on these networks end users can setup light paths through the networks on demand. Because light paths can be used to quickly move large amounts of data, or to get a guaranteed fixed quality of service regarding bandwidth delay. The goal is to have light paths in which network connections are created ad-hoc and reliably when needed. To achieve this, there are two important aspects that still need 1)a true dynamicity in the setup of paths 2) the extension of the path beyond a single domain to a true multi-domain setup. Therefore Light paths need to be consistently provisioned across network domains so that applications can effectively communicate end to end. Interoperability of provisioning systems requires communication of network information among them. Network topologies, network capabilities, scheduling information, and much more, need to travel between domains. The focus is on the definition of a standardized model for network information exchange to be used in hybrid networks. The model will primarily help provisioning systems in the setup of light paths. Nonetheless, the results of the work should be suitable for any other service within a hybrid network that deals with dynamic path computation. In this paper we propose a Multilayer Layer Resource Description Schema (MLRDS) based on Network Description Language intended for Multilayer networks which describes network topology in a machine readable format. MLRDS provides a common semantic to the application, the network and service provider, so that the communication between them is unambiguous. It can be used to create inter-domain network graphs at various abstraction levels, to provide an information model for service discovery and to facilitate light path provisioning.

The paper introduces the model topology description schema. The schema allows one to create parameterized descriptions of regular structures. The model description is modular: the user builds new module types out of simpler ones. The key idea of the schema is that properties of the topology (e.g. number of nodes, interconnection structure) should be able to be controlled by parameters. To facilitate this idea, sub module vectors and gate vectors were introduced. Parameters can be used to specify sub module types, sizes of sub module and gate vectors and also to describe multiple (or loop) connections. Using this schema, one can create regular structures such as star, chain, binary tree, hypercube etc which are the components of multilayer networks in an easy way. The concept of topology templates was introduced which is a tool for reusing interconnection structure. Three general patterns were presented that make it possible to generate any topology that can be described by mathematical formulas, including random topologies. The implementation of the schema allows one to easily create parametrized structures at run-time. The schema uses simple constructs which make it easy to implement. The schema is flexible enough to enable one to control the number and type of elements and their interconnection structure by means of parameters which is shown in Fig.1.

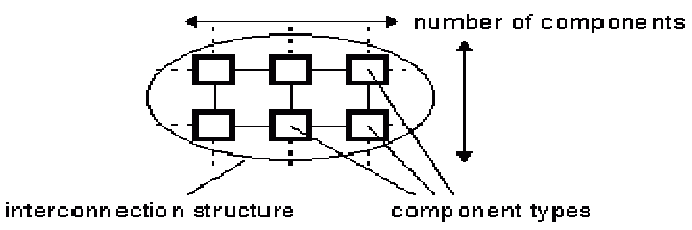

Fig.1. Aspects of the topology that should be allowed to be specified by parameters

The MLRDS schema intends to be such a description language.

The rest of the paper is organised as follows: Section II The problem statement III MLRDS Schema Structure IV Components of MLRDS Schema & Multilayer Connections V MLRDS reusable templates VI Conclusion is given.

II. Problem Statement

In order to integrate dynamic light path provisioning it is mandatory to support automatic topology discovery and path finding for the connections spanning multiple network domains. These domains must be able to exchange the required information expressed in an interoperable format to be able to understand the complexity inherent with light path request.

III. MLRDS Schema Structure

Schema consists of hierarchically nested modules which communicate with messages. MLRDS models are referred to as networks. The top level module is the system module. The system module contains submodules, which can also contain further submodules as shown in Fig.2. The depth of module nesting is not limited; this allows the user to reflect the logical structure of the actual system in the model structure.

Modules that contain submodules are termed compound modules, as opposed simple modules which are at the lowest level of the module hierarchy. Simple modules contain the algorithms in the model and they are implemented by the user.

Both simple and compound modules in a given network are instances of module types. While describing the model, the user defines module types and uses them to define more complex module types. Finally, the user creates the system module as an instance of a previously defined module type; all modules of the network are instantiated as submodules and sub-submodules of the system module.

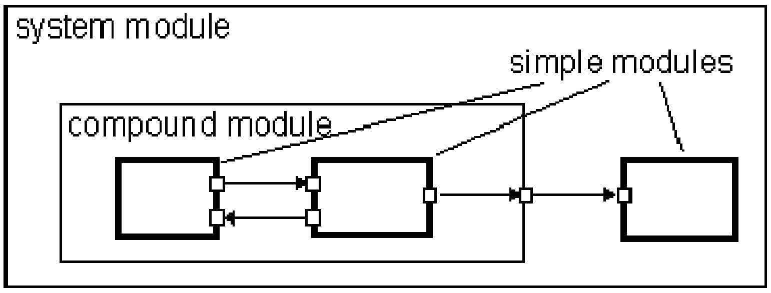

Fig.2. Model structure in MLRDS: compound and simple modules, gates, connections

In an MLRDS model, modules communicate by exchanging messages. Messages are sent out and arrive through gates, which are the input and output interfaces of a module.

Input and output gates of different modules can be interconnected. Each connection is created within a single level of the module hierarchy: within a compound module, one can connect the corresponding gates of two submodules, or a gate of one sub module and a gate of the compound module (Fig.3).

Due to the hierarchical structure of the model, messages typically travel through a series of connections, to start and arrive in simple modules. Compound modules act as 'cardboard boxes' in the model, transparently relaying messages between their inside and the outside world.

Modules can have parameters. Parameters are used for two main purposes: (1) to customize simple module behaviour, and (2) to parameterise model topology. Compound modules can pass parameters or expressions of parameters to their submodules.

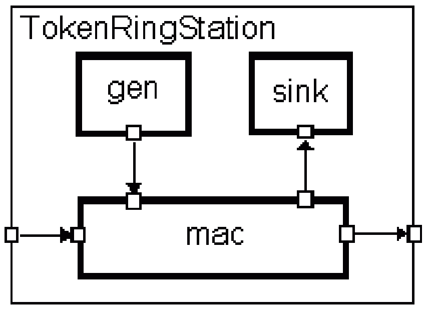

Fig.3. Submodules can be connected to each other or to the parent module

Numeric-valued parameters can be used to construct topologies in a flexible way. Within a compound module, parameters can define the number of submodules, number of gates, and the way internal connections are made.

IV. Components of MLRDS Schema

MLRDS Schema is defined by a textual network description. A network description contains declarations of simple module types, describes compound module types and contains a network definition that instantiates a compound module.

A simple module is defined by listing its parameters and gates:

simple Token Ring MAC
parameters:
THT, address;
gates:
in:from_higher_layer,
from_network;
out: to_higher_layer, to_network;
endsimple

Compound modules are modules composed of one or more submodules. Submodules can be either simple or compound modules. A compound module description - in addition to parameters and gates - also specifies the submodules and the connections within the module. For example, the compound module in Fig.3 can be defined by following code:

module Token Ring Station
parameters:
mac_address;
gates:
in: in; **out**: out;
submodules:
mac: TokenRingMAC
parameters:
THT=0.010,
address=mac_address;
gen: Generator;
sink: Sink;
connections:
mac.to_network --> out,
mac.from_network <-- in,
mac.to_higher_layer --> sink.in,
mac.from_higher_layer <-- gen.out;
endmodule

Multilayer Connections

If sub module vectors and/or gate vectors are used, the language allows one to create more than one connections with one statement. This is termed a multiple or loop connection. Multiple connections are created with the for statement in the connections section of a compound module description. One can place several connections in the body of the for statement, separated with semicolons. for statements can also be nested; this is done by specifying more than one indices in a for statement, with their own lower and upper bounds. Typical basic uses of multiple connections is to create one-module-to-many connections and to connect modules into a chain.

Sub module vectors, gate vectors and multiple connections are illustrated in the following example:

```
simple Hub
gates:
out: outport[];
endsimple
simple Station //...
module Star
submodules:
hub: Hub
gatesizes: outport[4];
station: Station[4];
connections:
for i=0..3 do
hub.outport[i] --> station[i].in;
endfor
endmodule
```

The result of the above is depicted in Fig.4.

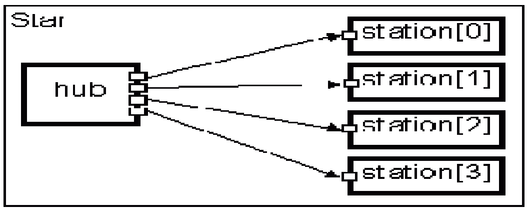

Fig.4. Multiple connections created using for

V. TOPOLOGY TEMPLATES

Leaving sub module type as parameter enables the user to create topology templates: compound modules that implement mesh, hypercube, butterfly, perfect shuffle and other topologies which are the components of multilayer networks, topology templates are a means of reusing interconnection structure.

The concept is demonstrated on a network with hypercube interconnection. When building an N-dimension hypercube, we can exploit the fact that each node is connected to N others which differ from it only in one bit of the binary representations of the node indices (Fig.5).

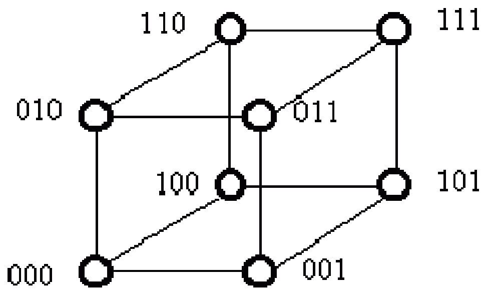

Fig.5. 3-D hypercube interconnection

The hypercube topology template is the following (it can be placed into a separate file, e.g hypercube.shm):

```
simple Node
gates: out: out[]; in: in[];
endsimple
module Hypercube
parameters:
dim, nodetype;
submodules:
node: nodetype[2^dim] like Node
gatesizes:
out[dim], in[dim];
connections:
for i=0..2^dim-1, j=0..dim-1 do
node[i].out[j] --> node[i # 2^j].in[j]; // # is bitwise XOR
endfor
endmodule
```

When the user creates an actual hypercube, he substitutes the name of an existing module type (e.g. Hyper cube_PE) for the node type parameter. The module type implements the algorithm the user wants to simulate and it must have the same gates that the Node type has. The topology template code can be used through file inclusion.

```
include "hypercube.ned"
simple Hyper cube_PE
gates: out: out[]; in: in[];
endsimple
network hypercube: Hypercube
 parameters:
dim = 4,
nodetype = "Hypercube_PE";
endnetwork
```

Another popular interconnection structure is the binary tree. The following topology template creates a binary tree of arbitrary height from nodes shown in Fig.6.

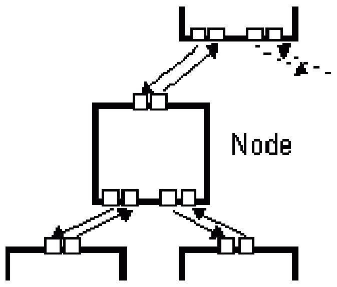

Fig.6. Binary tree node

Nodes of the binary tree are mapped to module vector indices like in the heap data structure: node[0] is the root element and node[i]'s children are node[2i+1] and node[2i+2].

```
module BinaryTree
// node gate indices: 0=up 1=left 2=right
parameters:
height, nodetype;
submodules:
node: nodetype[2^height-1] like Node;
gatesizes:
out[3], in[3];
connections nocheck:
for i=0..2^(height-1)-1 do
node[i].out[1] --> node[2*i+1].in[0];
node[i].in[1] <-- node[2*i+1].out[0];
node[i].out[2] --> node[2*i+2].in[0];
node[i].in[2] <-- node[2*i+2].out[0];
endfor
endmodule
```

VI Conclusion

In this paper we have proposed a schema, which can create regular structures such as star, chain, binary tree, hypercube etc. in an easy way which is capable describing the survivable multilayer network. The concept of topology templates was introduced which is a tool for reusing interconnection structure. Two general patterns were presented that make it possible to generate any topology including random topologies. At present we have not worried about the specific compiler to compile the language but in future we plan to develop a compiler which can run on different platforms.

REFERENCES

1. Ramesh K;H.B.Walikar"Algorithmic Design for Multi-Layer Survivable Networks and Failure Recovery," International Journal of Engineering and Innovative Technology (IJEIT) Volume 3, Issue 2, August 2013.
2. H.B Walikar;Ramesh K; Hanumantu K-Edge-Connectivity: a new Approach of finding Minimum Spanning Tree Ordered Minimum Spanning Tree (OMST) ," International Journal of Engineering and Innovative Technology (IJEIT) Volume 3, Issue 2, August 2013.
3. Jeroen van der Ham;Paola Grosso"Using the Network Description Language in Optcal Networks."
4. Cigno, R.L.; M.Munafo. 1995. "RC - A Flexible Language for the Specification of ATM Networks Simulation Experiments." In Proceedings of the Third Workshop on Performance Modelling and Evaluation of ATM Networks. (Ilkey, UK, July 2-6).
5. Marsan, M.A.; A.Bianco; T.V.Do; L.Jereb; R.L.Cigno; and M.Munafo. 1995. "ATM Simulation with CLASS." Performance Evaluation 24: 137-159.
6. MIL 3, Inc. 1996. OPNET Modeling Manual, Release 3.0, Washington, DC.
7. Moldovan, D.I. 1993. Parallel Processing, from Applications to Systems, Morgan Kaufmann, 191-204.
8. Shanmugan, K.S.; V.S.Frost; W.LaRue. 1992. "A Block-Oriented Network Simulator (BONeS)." Simulation. (Feb.).
9. Varga, A.; Gy.Pongor. 1997. "Flexible Topology Description Language for Simulation Programs". In Proceedings of the 9th European Simulation Symposium. (Passau, Germany, October 19-22).