STARS: A SIMULATOR FOR PERFORMANCE STUDY OF AGGREGATION BASED HIERARCHICAL ROUTING

Baruch Awerbuch[*] *Yi Du*[†]
Department of Computer Science
Johns Hopkins University
Baltimore, MD 21218
{baruch, yidu}@cs.jhu.edu

Yuval Shavitt
Bell Laboratories, Lucent Technologies
101 Crawfords Corner Rd., room 4G-627
Holmdel, NJ 07733-3030
shavitt@lucent.com

ABSTRACT

In this paper we describe STARS, the Scalable Topology Aggregation Routing Simulator, for performance study of aggregation based hierarchical routing that is proposed as a scalable routing solution for future large size networks. Adopted by ATM PNNI standards, aggregation based hierarchical routing approach maintains summary (or aggregation) information of lower level topologies at hierarchy nodes.

The main components of STARS are simulation engine, topology generator, and traffic generator. Besides ATM PNNI functionality, it implements reusable libraries for event scheduling, hierarchy generation and traffic generation. Routing is performed by shortest path algorithms with certain cost metrics in simulation engine. An event-based approach generates traffic loads at various granularity levels. STARS essentially realizes an architecture for hierarchical information exchange and maintenance, and provides interaction mechanisms among hierarchy entities usable by other simulations. STARS is used for performance study of ATM PNNI routing on link cost metrics, aggregation schemes and re-aggregation policies.

1 INTRODUCTION

Routing through a network can be accomplished if each network node maintains current routing information for the entire network. For future large scale networks with huge number of interconnected nodes grouped into subnetworks called *domains*, it is no longer practical that each node maintains full routing information due to large network size. Moreover, the dynamic nature of networks prohibits efficient information update for large networks. Routing with full information available at every node is not feasible.

Hierarchical source routing is proposed to cope with the scalability problem associated with large networks. This approach groups a network into hierarchy of nodes at various levels. Every node would maintain information on the topology underneath itself and exchange information with other nodes. The result is a local view of the network maintained at each node. Routing algorithms such as shortest path algorithms are used by nodes to make routing decisions based on locally maintained information. ATM PNNI [(ATM Forum 1996)] routing standards and Internet Nimrod [(Castineyra *et al.* 1996)] routing protocols both adopt this hierarchical routing strategy.

Typically a domain corresponds to a management organization or a geographical region. Internal structure details for a domain are usually subject to hiding due to either scalability or security reasons [(ATM Forum 1996), (Castineyra *et al.* 1996)]. Instead of exposing full domain structure, a summary, or *aggregation* of domain routing information is reported to outside nodes. This process of aggregating topology routing information, called *topology aggregation*, is essential for reducing maintained information for scalable routing performances in large networks, as the complexity of network routing information grows linearly with the number of network links for both source routing and link-by-link routing. Topology aggregation can be performed by many schemes with trade-offs on representation size and representation accuracy of routing information. Among these schemes are minimum spanning tree (MST) scheme, probabilistic tree scheme [(Bartal 1996)] and spanner scheme [(Althofer *et al.* 1993), (Peleg and Schäffer 1989)].

In this paper, we discuss the design and implementation of STARS, the Scalable Topology Aggregation Routing Simulator, for performance study of aggregation based hierarchical routing. ATM PNNI standards [(ATM Forum 1996)] are used as representative of the

[*] also with MIT Lab. for Computer Science. Supported by ARPA/Army contract DABT63-93-C-0038, DARPA/Airforce contract F30602961029, NSF Award ID CCR9700157, and ARO contract #DAAH04-95-1-0607.

[†] Corresponding author. Supported by DARPA/Airforce contract F30602961029.

approach, mainly because PNNI routing and signaling standards are relatively mature and performance study on PNNI routing would have significant implication on ATM network deployment. STARS emulates ATM PNNI routing to study its performance on aggregation schemes and link cost metrics.

The rest of the paper is organized as follows. Section 2 introduces hierarchical source routing and the main goal for our simulation study. In section 3, we discuss specific design and implementation of STARS. In section 4 STARS is used for performance study of ATM PNNI routing with various aggregation schemes and link cost metrics. Section 5 gives the conclusion.

2 BACKGROUND

In this section, we consider hierarchical source routing and goals for our simulation performance study.

2.1 Hierarchical Source Routing

In *source routing*, link state information are maintained by network nodes for routing purposes. Routing decisions are made by source nodes that receive connection requests, using locally maintained routing information. Routing information for links is usually presented to routing protocols as a value of certain cost metric. Routing algorithms use the link cost metric values to determine feasible paths for requested connections. Shortest path algorithms such as Dijkstra's Algorithm are widely used for this purpose. When information for all physical links is maintained at every node, routing algorithms can easily determine whether there are feasible paths for requested connection, and if so the best path. Additional steps are needed to reserve bandwidth for all links on the found path.

When network becomes very large, it is not feasible to demand each node maintain full information for all physical links. Instead, a hierarchy upon physical networks is imposed to obtain nodes at various levels. A node can maintain processed, or *aggregated*, information about the network, which is not necessarily full information about nodes and links for entire networks. Nonetheless, source node for a connection request can still find a sequence of partial paths, each with certain levels of abstractions in the hierarchy. When realized by physical links, these partial paths combine to present a physical path for connection request. The successful establishment for each partial path in this sequence realizes a route of physical links for the request. This routing style involves routing at hierarchical levels and is called *hierarchical source routing.* It is the approach adopted by ATM PNNI standard [(ATM Forum 1996)]. After the partial paths are determined, a *signaling* process is invoked to traverse links on these path to reserve bandwidth on those links that have enough bandwidth to support the request. When a link can not support a request, a backtrack (or *crank-back*) action makes the traverse back to the previous node and try another link from there.

Routing information advertised by a node to others is aggregated by *topology aggregation scheme* to reduce routing information storage in nodes and topology information exchange, the main reason why hierarchical routing is scalable. Topology aggregation essentially derives a simpler connectivity graph out of a much more complicated real connectivity graph. Various topology aggregation schemes exist, among them are tree scheme [(Bartal 1996)] and spanner scheme [(Althofer *et al.* 1993), (Peleg and Schäffer 1989)]. How aggregation schemes affect performance for hierarchical routing is one focus of our simulation.

2.2 Simulation Goals

The goal of our simulation is to study the performance of topology aggregation based hierarchical source routing. In particular, we study the performance of aggregation schemes and link cost metrics in the hierarchical setting. Two performance measures are of more interest. One measure on the realized connections from all connection requests is called *network throughput*, which is the ratio of network traffic flows for realized connections to that of the total connection requests. Another measure about the signaling process associated with the route establishment for connections is called *signaling delay*, which considers the number of crankbacks encountered during the signaling process and the associated time the signaling process takes. Our simulator is to emulate the ATM PNNI routing. We want the simulator to be flexible enough to adapt to other applications as well.

There exist several simulation granularity levels. One granularity is on packet level, where packet switching is simulated. This simulation level is very time-consuming despite the relatively accurate performance measurements. Simulation at higher simulation levels, such as traffic flow, could shed more insights in a shorter time. Higher level simulation could lead to identification of important system parameters that affect performance the most. High level simulation can be used to identify important parameters qualitatively, while detailed simulation could verify findings with more accurate quantitative performance measurements. A good simulator is capable of supporting multiple simulation granularity levels.

Hierarchical source routing such as ATM PNNI standards is still under development. Not many simulators emulate PNNI functionality. Most of the simulators developed for network performance evaluation are specific to their applications and thus difficult to adapt to our purposes. Besides, generation for hierarchy based network graphs, an essential part for hierarchical source routing simulation, is not well studied.

3 THE SIMULATION ENVIRONMENT

In this section, we discuss the design and implementation of STARS, the Scalable Topology Aggregation Routing Simulator, for performance study of hierarchical routing. We focus on general structure of STARS, i.e., the general mechanisms realized by STARS on control passing and information exchange among nodes. Section 4 is dedicated to the aspects specific to ATM PNNI routing performance studies.

STARS is an object-oriented discrete event simulator in C++ that emulates ATM PNNI routing protocols on hierarchically structured networks. It adopts a modular architecture that enables easy integration of modules for various capabilities. For example, a generic interface for link cost metrics is provided for easy plug-in of cost metric functions, and interfaces are provided for topology aggregation schemes. STARS also supports multiple granularity of network traffic.

The architecture of STARS is depicted in figure 1. It consists of the following components.

- **Simulation Engine**: the engine supports graph representation of topology, provides dynamic cost metric update mechanism over the topology, provides mechanism to record performance statistic of aggregation schemes;
- **Topology Aggregation Server**: the server implements various aggregation schemes such as trees and spanners;
- **Network Topology Generator**: the generator produces specified structure of domain topology;
- **Network Traffic Load Generator**: the generator produces network traffic (connection requests) to drive simulation engine;
- **Statistics Collector**: this component collects information used by traffic statistics analyzer;
- **Graphical Display**: the display shows routing signaling progress graphically;

Simulation engine uses shortest path algorithm with certain link cost metrics to select feasible paths for requested connections. This routing formulation is powerful since many existing routing schemes fit well in it. For example, the Internet min-hop routing can be realized by a shortest path algorithm with simple constant link cost. More complicated link cost functions that take into account other properties such as link utilization and link capacity capture more link status information and thus promise better description of the current network status. One such metric is the exponential cost metric obtained as an exponential function of current link utilization. Theory suggests there exist on-line routing algorithms based on exponential costs that could achieve optimal performance

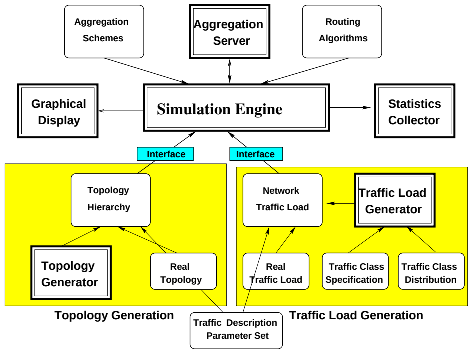

Figure 1: The STARS Architecture

of throughput against off-line algorithm. Our study focus on performance of exponential cost metric and that of constant link metric in hierarchical routing. To facilitate comparisons for various cost metrics, STARS allows easy incorporation of link metric functions.

Aggregated domain information needs to be recalculated when changes on topology parameters occur to justify aggregation on topology information. **Aggregation server** accepts as inputs the current network topology and desired aggregation scheme from network nodes, and returns aggregated cost metric to the calling nodes as aggregation calculation result using the specified inputs. Many aggregations may be called by different nodes concurrently. As a result, besides providing capability for direct calls from nodes to aggregation functions that are executed sequentially on the same machine, we provide an option to run several aggregation servers simultaneously on several machines to alleviate loads on one single machine for increased performance.

Network topology generator produces various hierarchical topologies to be studied, and **network traffic load generator** produces dynamic connection requests as inputs to STARS to drive the simulation. A request is represented by its staring time, source and destination node, and traffic bandwidth requirement. We use a simple topology description format and allow input to specify connection request streams with various arrival/departure distribution and bandwidth requirement. **Statistic collector** is used to collect data of interest and perform some simple analysis, and **graphical display** shows routing signaling progress for the current connection requests under establishment on a graphical terminal.

STARS take as input system parameters such as link cost metric functions, aggregation schemes, hierarchy topology, and traffic load. System parameters can be set in STARS configuration file. STARS also accept command line inputs for choices on the system parameters. In the following, we discuss implementations of these components in more detail.

3.1 Simulation Engine

Simulation engine adopts an event-entity model. Event happens at certain discrete time. Active system parts such as network nodes respond to messages associated with an event by performing appropriate actions and generating new events (with messages). General event format containing messages is provided to facilitate message exchange among active parts.

Each active part is modeled as an entity that would receive messages, process them by performing its own actions based on the received messages, and send new messages to other entities via events. Interaction among entities is emulated through events, as messages sent by active parts are embedded on events that are time-stamped with the time when message receiver is to receive and process the message.

We implement an event-handling library to process general event manipulation (insertion/removal). This library contains Event and Entity, as well as EventScheduler classes. Event class contains the following information: occurrence time for this event, sender that generates the event, receiver that receives the event, and messages passed from sender to receiver. It is quite general as it does not impose specific message formats. User-defined messages can be easily incorporated into Event instances for exchange. EventScheduler class maintains events into a priority queue ordered by the event occurrence time.

Entity class models active parts such as network nodes. Since entity would receive messages from other entities through events, process them and send new messages to other entities, a general mechanism for entity interaction is provided by Entity to perform the functionality. Application-specific parts are realized as derived classes of Entity, which use the inter-entity message exchange mechanism provided by Entity. In general, we can easily implement new entities on top of Entity class for simulations using event-entity model.

For ATM PNNI simulation, we implement PeerNode class on top of Entity class, which is specific for hierarchical topology aggregation routing. PeerNode represents nodes in topology hierarchy at various levels with corresponding event-handling functionality.

3.2 Aggregation Server

To perform topology aggregation, a simple method for invoking aggregation schemes, called *direct aggregation calling*, is provided for nodes to direct call aggregation functions within the single simulation process. All aggregations are performed in that process. This approach is hardly scalable since when networks become bigger, the time for topology aggregations would increase considerably.

Nodes in PNNI hierarchy needs to perform topology aggregations for updated topologies in a timely fashion. Ideally we can achieve this requirement if computer host can sustain the aggregation processing in real time, which would not be the case when simulating large size networks. One solution is to use more processes to alleviate aggregation processing for the simulation process. Aggregations can be performed by processes running on other hosts. Simulation could run much faster as processes from other hosts would relieve aggregation burdens from the simulation engine. Aggregation servers are used to speed up the aggregation response time.

Server-based aggregation calling method uses aggregation servers to implement topology aggregation schemes. Every node sends its aggregation requests as messages containing aggregation scheme type and its current topology to aggregation server that would perform the aggregation according to the request and send results back to the node when it is done. Communication between nodes and servers is realized by TCP/IP sockets. One server can serve aggregation requests from other hosts with socket interface. These aggregation servers can be used as stand-alone solution for topology aggregation processing for remote nodes. Several aggregation servers can run concurrently to serve clients and speed up the simulation.

3.3 Network Topology Generator

Topology generation involves producing both physical-level topology graph and a hierarchy imposed on this graph. Most researches on graph topology generation focus on one level connectivity among nodes. There is limited research on generating hierarchical structure of physical topology.

We implement routines to produce hierarchy-imposed topologies, where routines generating topologies at one level are used together with routines that construct hierarchies from these one-level topologies.

The routines that generate one level graphs involving no hierarchy are called *graph generators*. Two classes of graph generators are provided. *Regular graph generators* produce regular topologies such as rings, regular graphs, and shuffle networks, while *random graph generators* produce random topologies with certain set of parameters. For example, our random generators can produce random graphs similar to that appear in [(Waxman 1988)].

The routines that construct hierarchies by combining one-level topologies are called *hierarchy constructors*. Given a high level structure among high-level nodes (which is an one-level topology itself) and topologies for high-level nodes (which are low-level graphs containing some *border nodes* to be connected to border nodes from other domains), hierarchy constructors connect the border nodes of high-level nodes to realize connectivity to ensure high-level nodes be connected as described in the high-level structure.

3.4 Network Traffic Generator

Traffic load on hierarchical topology graphs drives simulation running. Real traffic trace can be used for this purpose to provides some realistic scenarios of traffic loads in real networks. Accordingly STARS provides the capability to allow real traffic traces to serve as traffic load inputs.

Real traffic traces are difficult to obtain, and they are usually not flexible to present more dynamic traffic load scenarios being obtained under certain conditions. To deal with this problem, traffic generation based on traffic models is implemented to provide classes of traffic loads. Due to lack of understanding of network traffic, realistic traffic generation remains an active research area. In our study, we generate connection requests with certain degree of abstraction based on queueing theory results.

A traffic generation library is implemented to generate connection requests, the traffic for our performance study, based on Event class. A connection request consists of starting time, duration, connection source and destination nodes, and desired bandwidth. Several distributions are implemented for connection arrivals and departures, as well as requested bandwidth. The traffic generation can be specified in STARS configuration file. For the current study, we simulate routing request at *bandwidth* level. That is, connections with certain requested bandwidth are used as traffic and simulations are conducted on the amount of bandwidth for these connections. Bandwidth reservation is deployed to ensure bandwidth availability for admitted connections. Requests can be rejected due to lack of available bandwidth.

We model request duration and bandwidth requirement using random variables of certain distributions. Request arrival time, however, could depend on previous requests. This means that inter-arrival time between consecutive requests is a random variable of certain distributions. Three variables are used to describe a sequence of connection requests, namely *Inter-arrival Time* between requests arrival, *Duration Time* for request duration, and *Required Bandwidth* for bandwidth requirement. Distributions currently available are constant, uniform and exponential distributions, with mean values. Each of the three parameters can assume any of these distributions.

We usually use exponential distributions for both inter-arrival time and duration time. There are two reasons for this choice. One reason is that this particular distribution combination presents a very irregular pattern of request arrivals, which is desired for performance study under random scenarios. The other reason is a queueing theory result gives a formula for network traffic load for this arrival pattern. Using this formula, we could have rough understanding of network loads when we choose distribution parameters for performance simulation studies.

A flexible scheme is adopted by the traffic library to control event generation from connection request specification. This capability is important since too much memory could be consumed if many connection requests are generated as events. When generating infinite requests stream, this capability is crucial as it is impossible to generate all requests. Our scheme only generates events for the near future.

We use an event-based approach for generating traffic load. An entity called Traffic is introduced to handle traffic generation. Events are sent to Traffic for new connection requests. The traffic library view traffic (requests) as a *stream*, which logically consists of consecutive connection requests. A stream is described by some characteristics such as starting time and distribution for requests variables. A stream can generate some requests and becomes a new stream with new values for its characteristics such as starting time. The request generation process is called *stream unfolding*. When receiving traffic-generation event, Traffic entity would *unfold* the current stream into connection requests to be determined by a *unfolding factor*, and a new stream is generated to reflect the current traffic state. An event for the new stream is passed to Traffic. The unfolding factor determines the number of active requests for STARS.

By using unfolding factor, this approach controls the number of generated connection requests, the basic units for simulation study that are dealt with by simulator entities. This prevents memory waste for requests not needed at this point, which is important for running large scale simulation. This would allow the simulation of infinitely long traffic. Appropriate value for unfolding factor, a changeable simulator parameter, would balance memory utilization and event processing for the traffic stream.

An interesting and powerful property of the traffic generation library is its unique way to achieve abstraction levels of traffic descriptions. Though the currently available parameters are not sophisticated, the library is capable of more detailed specification and generation of connection requests. A *traffic load hierarchy* is provided by the event-based approach. Each generated event for connection request, which now specifies a bandwidth requirement, can be viewed as a *traffic flow* in some time interval with certain degree of abstraction. The flow can be associated with a sequence of packets. If the event-handling for this event is implemented, more concrete traffic can be generated as input to simulators on finer granularity levels. A simple example is a two hierarchical traffic model consisting of traffic flows and packets. Flows are first generated as events at higher levels; when flow events are processed, packet-level traffic loads are generated by event-handling procedures. If desired, more traffic granularity levels can be realized, depending on the applications and goals of the simulations.

4 SIMULATION: ATM PNNI

We use STARS to study the performance of ATM PNNI routing on aggregation schemes and link cost metrics, two important parameters for ATM PNNI routing framework [(Awerbuch *et al.* 1998)]. Other system parameters include network topology and traffic loads and are inputs to STARS. Network performance is measured by network throughput defined as the total flow successfully transmitted by network, and control overhead such as signaling delay that measures efficiency of routing establishment.

4.1 Link Cost Metric

We consider two link cost metrics, namely, constant metric and exponential metric.

constant — The cost for each link is a constant regardless of the currently available bandwidth on the link. This corresponds naturally to minimum hop routing in the Internet.

exponential — Link cost is an exponential function of the remaining bandwidth on the link. This cost function is justified by both queueing theory suggestion and opportunity cost functions.

4.2 Aggregation Schemes

Aggregation schemes play an important role in network performance. Various aggregation schemes would incur different computation requirement and representation size. The aggregation schemes available in STARS are listed below.

- Complete : No aggregation is done. The full cost matrix between border nodes is used.
- Diameter: A star aggregation where the cost of the star radius is half that of network diameter.
- Average: A star aggregation where cost is half the average cost of distances between border nodes.
- MST : A minimum spanning tree representation.
- RST: A random spanning tree.
- Spanner: A t-spanner.

4.3 Network Dynamics

Besides link cost metrics and aggregation schemes discussed in the above, there are other parameters that would also affect network performances. They include hierarchical network topology, link capacity, link delay and network load.

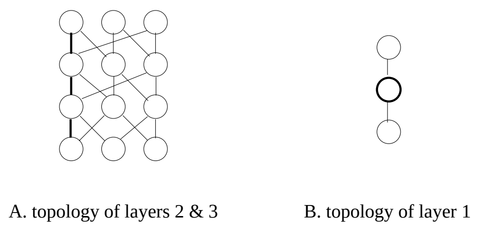

Figure 2: Self-similar Hierarchical structure. The middle node in layer 1 (right hand side) is structured as depicted on the left. In layer 2 each of the 12 nodes is structured recursively the same as a shuffle-exchange. Link capacities: 5 for internal links in layers 2 and 3, 10 for the bold internal links in layers 2 and 3, 10 for links between border nodes, 15 for links between border nodes that both have bold links.

4.4 Performance Measures

STARS considers two performance measures.

Throughput measured by the cumulative length of the realized connections, and

Control overhead measured by the average number of backtracks per connection. It indicates routing efficiency under certain aggregation scheme.

4.5 Simulation Results

Using STARS we study the performance of aggregation schemes and link cost metrics for ATM PNNI routing [(Awerbuch *et al.* 1998)]. We present result for one topology called *self-similar hierarchical (SSH)* depicted in Figure 2, which contains several levels as a switching network (shuffle-exchange).

Consider the performance of exponential cost metric against that of constant metric. Figure 3 shows the ratio of throughput between exponential and constant metric, and Figure 4 shows the ratio of crankbacks between the two metrics. We observe that exponential metric achieves more than 50% (sometimes more than 100%) gain in network throughput against constant metric, while suffers less than 20% of crankbacks.

Next, exponential metric is used since it performs much better than constant metric. We study the performance of various aggregation schemes listed in Section 4.2 for *SSH* topology. Figure 5 and Figure 5 depict throughput and crankbacks results respectively. We see that MST (minimum spanning tree), together with Spanner, RST and Complete performs much better than other schemes. Complete does not actually perform aggregation and is included here for comparison purposes. Our other studies shows that MST almost always performs better than other schemes with representation reductions.

Detailed results are in [(Awerbuch *et al.* 1998)].

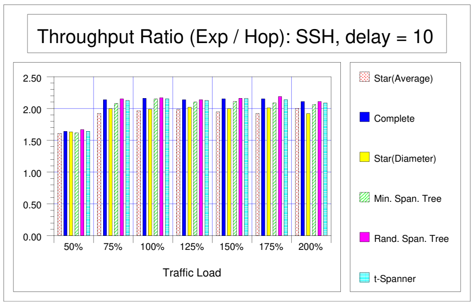

Figure 3: Ratio of throughput between exponential and constant cost metric for topology in figure 2

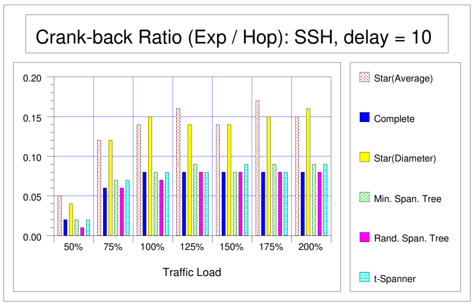

Figure 4: Ratio of crankbacks between exponential and constant cost metric for topology in figure 2

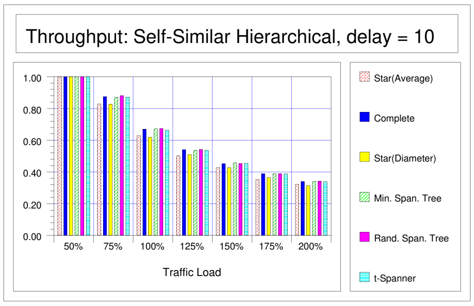

Figure 5: Throughput for various aggregation schemes with exponential cost metric for topology in figure 2

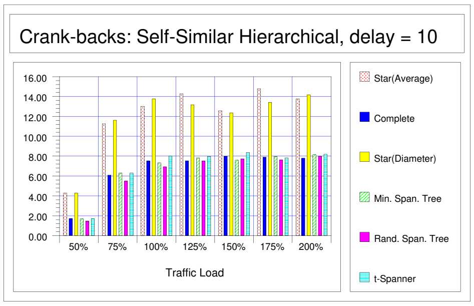

Figure 6: Crankbacks for various aggregation schemes with exponential cost metric for topology in figure 2

5 CONCLUSION

In this paper, we describe STARS, a simulator for studying the performance of hierarchical source routing with particular focus on aggregation schemes and link cost metrics. STARS adopts an object-oriented discrete event-based implementation. In the heart of STARS is simulation engine for routing decision making that essentially establishes an event-based mechanism for interactions among entities. Topology generator and traffic load generator produce hierarchy-imposed topology and network traffic in the form of network connection request stream. An event-based hierarchical description for traffic loads is deployed in traffic generation. STARS essentially realizes an architecture for hierarchical information exchange and maintenance, and provides interaction mechanisms among hierarchy entities usable by other simulations.

We have used STARS to study the performance of ATM PNNI routing. Our findings are that link cost metrics usually play significant role in network performance. There exist several aggregation schemes that consistently perform better than other schemes for both throughput and signaling delay.

References

I. Althofer, G. Das, D. Dopkin, D. Joseph, and J. Soares. 1993. On sparse spanners of weighted graphs. *Discrete and Computational Geometry*, 9:81 – 100.

Baruch Awerbuch, Yi Du, Bilal Khan, and Yuval Shavitt. 1998. Routing through networks with hierarchical toplogy aggregation. *3rd International Symposium on Computer Communications (ISCC'98).*

Yair Bartal. 1996. Probabilistic approximation of metric space and its algorithmic applications. In *37th Annual IEEE Symposium on Foundations of Computer Science.*

I. Castineyra, J. N. Chiappa, and M. Steenstrup. 1996. The nimrod routing architecture, Internet Draft, Nimrod Working Group.

David Peleg and Alejandro A. Schäffer. 1989. Graph spanners. *Journal of Graph Theory*, 13(1):99 – 116.

ATM Forum technical committee. 1996. Private network-network interface specification version 1.0 (PNNI), af-pnni-0055.000.

Bernard M. Waxman. 1988. Routing of multipoint connections. *Journal on Selected Areas in Communications*, 6:1617 – 1622.