ATM Support on NS*

Sowmya Manjanatha and Radim Bartoš

Dept. of Computer Science, University of New Hampshire, Durham, NH 03824, USA
{sowmyam,rbartos}@cs.unh.edu, Phone: (603) 862-3792, Fax: (603) 862-3493

Abstract

NS is a discrete event simulator that originated from the REAL Network Simulator [?]. NS focuses on a wide range of protocols such as TCP/IP, OSPF, and BGP. In addition to several protocol investigations, NS is used as an extensive research tool in areas of multicast transport, multimedia, and wireless networking. NS implements multiple levels of abstraction to provide improved simulator performance. The top level abstraction is a two-level simulation evironment. A high-level scripting language OTcl [?] is used to provide a simulation interface and a traditional compiled programming language (C++) is used for the actual simulation itself. The idea behind multi-level abstraction is the scalability of simulation. Users can trade simulator performance for packet-level accuracy by adjusting the level of simulator abstraction. Although increasing the simulator abstraction level increases modularity, decreasing it results in a more realistic simulation. NS allows the flexibility of adjusting the level of abstraction. NS has been written in a highly object-oriented fashion using the object- oriented features of C++ and OTcl.

1 NS Architecture

Figure 1 depicts a high-level NS architecture. NS is designed such that the Simulation/Execution Module and the Configuration Module are strictly decoupled from each other. This maintains transparency to the user in addition to providing the ease of configuring without having to delve into complex simulation details. The Simulation Module includes the Scheduler and the Handler. The main task of the Scheduler is to schedule events provided by the configuration

*This research was supported in part by NSF-ARI grant No. 9601602.

module for simulation. The handler dispatches the individual events to the appropriate simulation module in simulation order. The Configuration Module consists of a set of network objects such as *Nodes, Links, Traffic Sources, Queues,* etc.

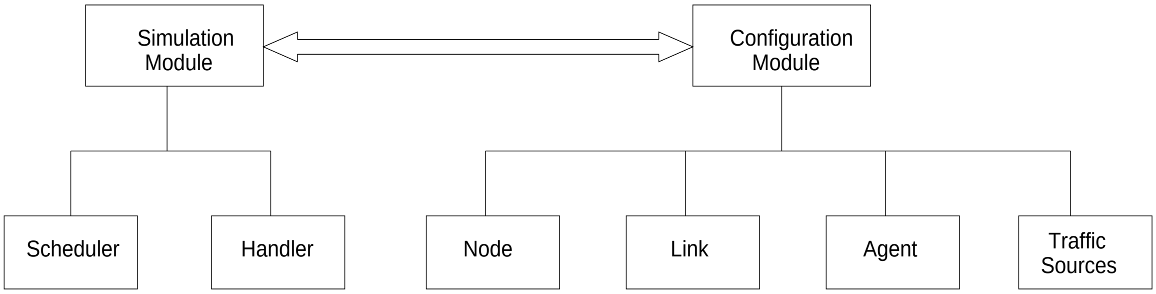

Figure 1: High Level NS Architecture

The *Simulation Module* and most of the *Configuration module* objects are written in C++. Each network object also provides a Tcl handle that enables the user to choose and combine them appropriately as the simulation needs. The network objects primarily represent the real-world networking equipment or one of the layers of the seven-layer OSI model. For example, the Nodes could either represent an IP router, ATM switch, bridge, etc. The Traffic Sources and Agents depict the application and transport layer elements respectively. In order to include the ATM simulation functionality, only the Configuration module had to be enhanced, keeping the Simulation module intact. The next section describes the design and implementation of the ATM functionality on NS.

2 ATM Architecture on NS

ATM is a connection oriented technology while the protocols that were available on NS were mostly IP based. Enhancing NS to simulate ATM therefore required implementing ATM equipment such as an ATM Switch, an ATM End Station and an ATM Link. The remainder of this Chapter describes the design and implementation of ATM simulation components on NS.

3 ATM Switch Design

Figure 2 depicts the design of an ATM switch and attached to it is an outgoing ATM Link. The design of the ATM Link is explained in section 3.4. The main function of an ATM switch is the forwarding of incoming data on to the next appropriate destination. The Switch Fabric shown in Figure 2 is responsible for the forwarding. The switch executes forwarding through control paths known in ATM terminology as Virtual Circuits (VC) and Virtual Paths (VP). The switch also performs certain traffic conditioning functions such as policing and shaping to provide QoS. The ATM Switch on NS therefore consists of three main modules, the *Connection Manager (CNXM)*, the Switch Fabric (SWFAB) , and the *ATM Conditioner.* In NS, All the four modules were added to the basic *Node* on NS to enhance it to an ATM Switch as shown.

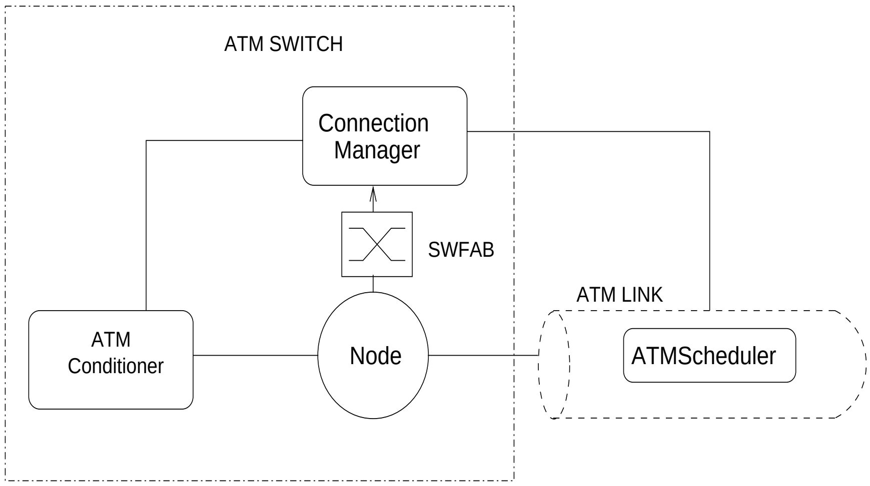

Figure 2: ATM Switch Implementation, SWFAB = Switch Fabric.

3.1 Connection Manager (CNXM)

The Connection Manager incorporates a PVC Database that holds information about all the active PVCs on the switch. CNXM also provides functions to *create,* and *delete* a PVC, and lookup PVC Database. Each entry on the PVC database is an object containing the VPI/VCI values of the incoming and outgoing PVCs and the traffic descriptors as negotiated with the user during Connection Admission. SVCs are not supported in the current version since PVCs provide all the functionality required for the IP DiffServ to ATM QoS mapping schemes. Figure 3 shows a simple PVC Database with a few example connection descriptors.

INCOMING PVC				OUTGOING PVC			
VPI	VCI	Traffic Descriptors		VPI	VCI	Traffic Descriptors	
		RXTP	TXTP			RXTP	TXTP
0	100	0x00abcd	0x00abfd	0	101	0x01bca9	0x02efc2
0	200	0x00affe	0x00ffee	0	201	0x01ce08	0x02ffc8
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figure 3: PVC Database.

3.2 Switch Fabric

The main function of the Switch Fabric is the forwarding of an incoming cell to the appropriate destination. Additionally, the Switch Fabric is responsible for updating the ATM header values in the cell as follows:

The incoming cell contains the VPI/VCI values of the PVC on which it arrives. The VPI/VCI values are used to obtain the corresponding outgoing PVC for the cell. The Switch Fabric replaces the VPI/VCI values of the cell with that of the VPI/VCI values from the outgoing PVC and forwards it to the appropriate destination.

3.3 ATM Conditioner

Figure 4 shows the components of the ATM Conditioner and the relationship between them.

The description of each of the ATM Conditioner components follows:

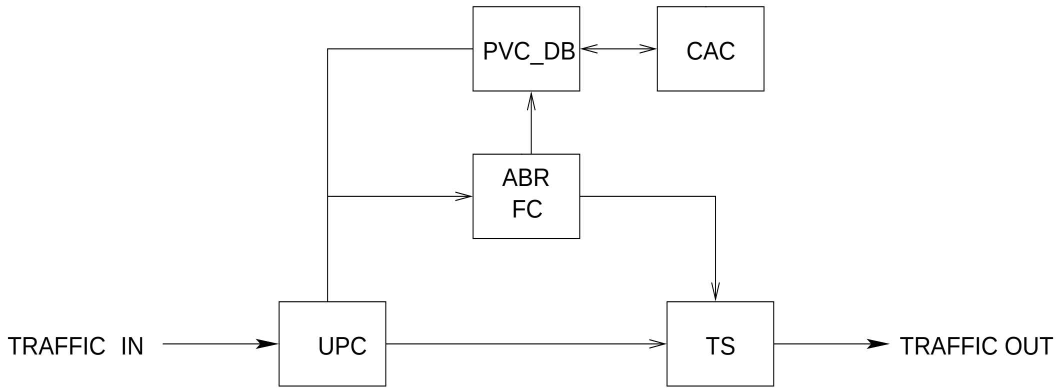

Figure 4: ATM Conditioner, FC = Feedback Control.

Connection Admission Control (CAC): CAC determines whether an incoming connection should be granted or rejected. The criteria for accepting or rejecting the connection depends on the type of connection requested. For example, a CBR connection is accepted only if the available bandwidth exceeds the PCR of the connection under consideration.

Usage Parameter Control (UPC): UPC is another name provided for traffic policing. UPC was implemented using the Leaky Bucket/GCRA algorithm [**?**] in which traffic is delivered out of the ATM interface at a rate requested by the user. Traffic coming in at a higher rate is stored till the buffer limit is reached. Any excess incoming traffic is then discarded.

Traffic Shaping (TS): Traffic Shaping is a function applied to smooth out bursts in the incoming traffic. TS was implemented using the Token Bucket algorithm as described in the ATM Forum Specification [**?**].

ABR Feedback Control (ABR FC): The Explicit Rate Indication Control Algorithm (ERICA) [**?**] was used to provide the Congestion Control mechanism required by the ABR service.

The traffic conditioning components described characterize the incoming traffic to adhere to the requested service criteria.

3.4 ATM Link

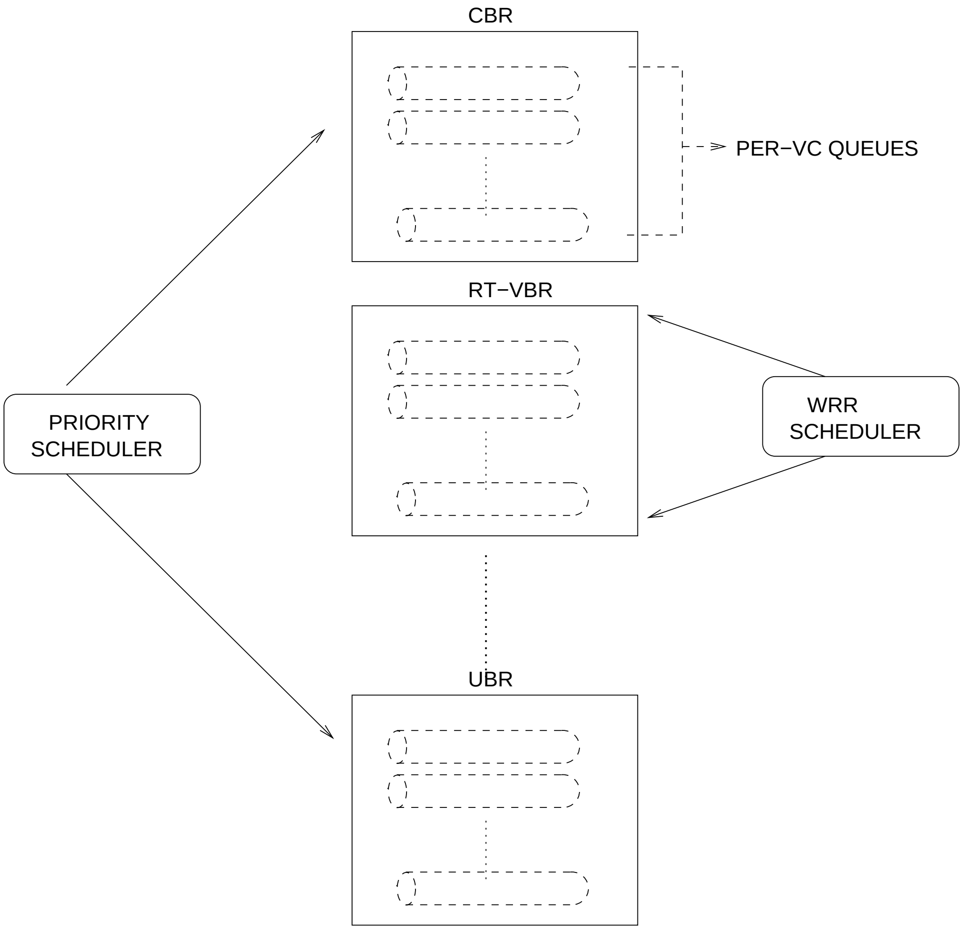

Figure 5: ATM Queuing and Scheduling.

The prioritization of traffic is made possible by the Queuing and Queue Scheduling mechanisms. In NS, the Queues are attached to links. Therefore, a new type of link, namely the ATMLink was created and the Queue Scheduler or the ATM Scheduler was attached to this link. Figure 5 presents the queueing and the scheduling mechanism adopted in the ATM Scheduler. A two level queueing scheme was adopted for the Queue Scheduler. On the high level, an abstract queue for each of the service categories supported was created. Each category queue object contained per-VC-Queues. Therefore, a 2 level queue scheduling mechanism was used. Priority was given based on category, e.g, CBR gets the highest (0), rt-VBR and nrt-VBR gets second and third priority respectively, ABR gets the next priority and UBR gets no priority over other categories. A Weighted Round Robin (WRR) Scheduler was adopted inside each abstract queue. The weights are configurable and defaults are as follows. The CBR weight is calculated as the ratio of PCR to total link bandwidth. The VBR weight is calculated with respect to SCR, and the ABR weight similarly with respect to MCR. In this implementation of the simulator, GFR was not supported.

The QoS functionality on the ATM Simulator is incomplete without the use of the ATM Link to the ATM Switch since the priority scheduling of traffic required for QoS functionality is implemented in the ATM Scheduler that is part of the ATM Link. Therefore, an ATM Switch must always be used with the ATM Link. However, the ATM Switch and the ATM Link were designed such that both of them could function independently.

3.5 ATM Edge Device Design

ATM Switches recognize packets of fixed lengths called *Cells*. Each Cell also includes an ATM header standardized by the ATM Forum. In order for the data from an IP network to be recognized by an ATM switch, the outgoing IP packets must be converted into cells. The ATM Forum recommends an interface layer known as the ATM Adaptation Layer (AAL) that is responsible for the Segmentation and Reassembly of the packets into cells. An AAL5 implementation was added to the ATM feature on NS. An AAL5 capable NS node was configured as shown in Figure 6.

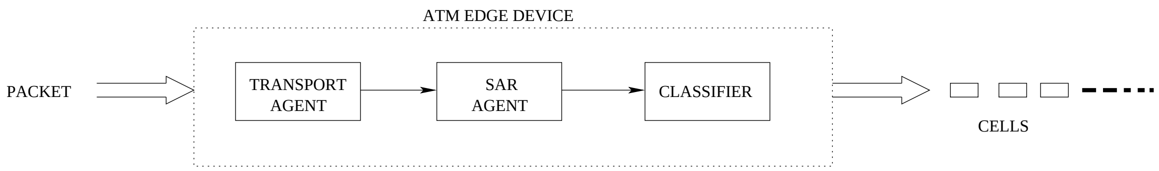

Figure 6: ATM Edge Device.

This Chapter introduced the Network Simulator (NS) used in our analysis of the mapping schemes and described the enhancements added to NS. The next Chapter discusses the simulations performed using NS and the evaluations deduced from the experiments.