COMP1047 – Computer Networks – Lab 2

Network Components

This section talks about the NS components, mostly compound network components. Figure 6 shows a partial OTcl class hierarchy of NS, which will help understanding the basic network components. For a complete NS class hierarchy, visit http://www-sop.inria.fr/rodeo/personnel/Antoine.Clerget/ns.

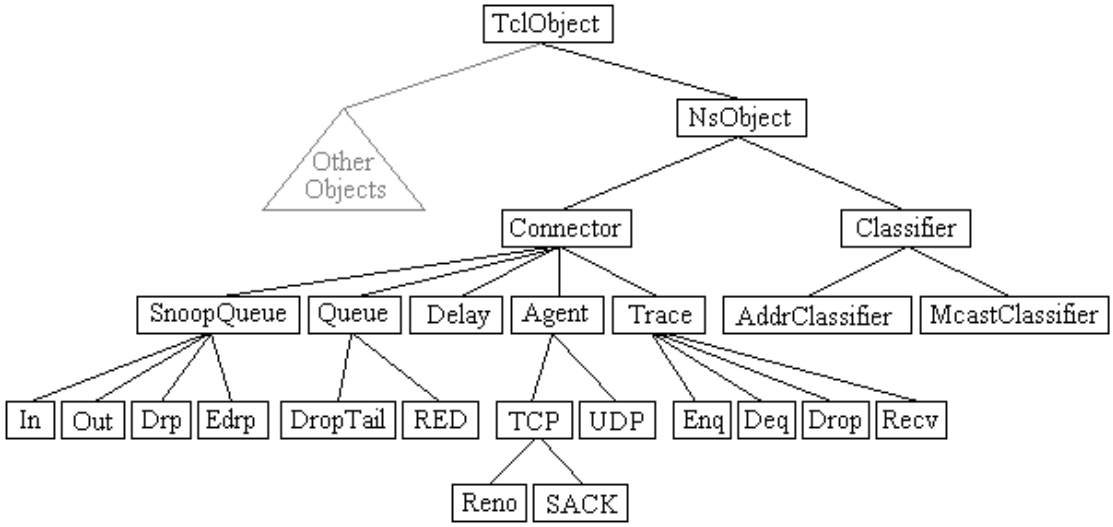

Figure 6. Class Hierarchy (Partial)

The root of the hierarchy is the TclObject class that is the superclass of all OTcl library objects (scheduler, network components, timers and the other objects including NAM related ones). As an ancestor class of TclObject, NsObject class is the superclass of all basic network component objects that handle packets, which may compose compound network objects such as nodes and links. The basic network components are further divided into two subclasses, Connector and Classifier, based on the number of the possible output data paths. The basic network objects that have only one output data path are under the Connector class, and switching objects that have possible multiple output data paths are under the Classifier class.

- **Node and Routing**

A node is a compound object composed of a node entry object and classifiers as shown in Figure 7. There are two types of nodes in NS. A unicast node has an address classifier that does unicast routing and a port classifier. A multicast node, in addition, has a classifier that classify multicast packets from unicast packets and a multicast classifier that performs multicast routing.

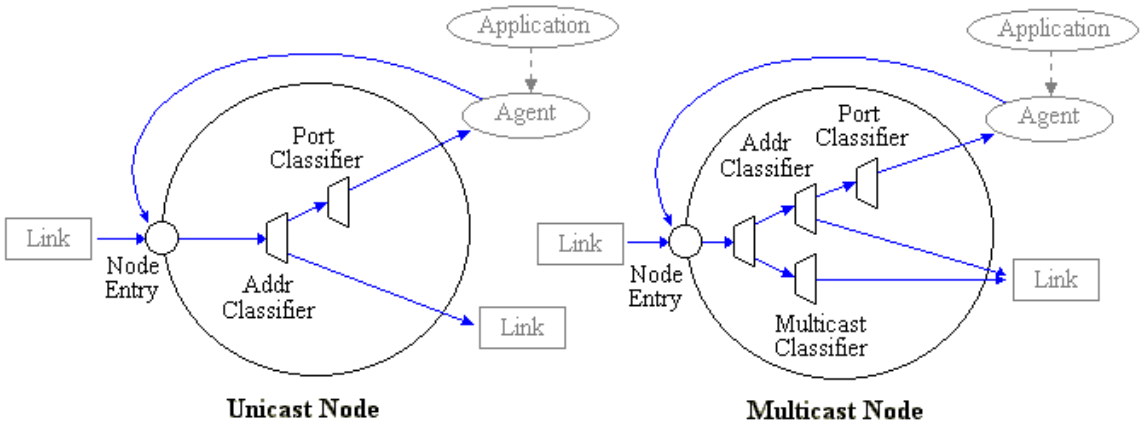

Figure 7. Node (Unicast and Multicast)

In NS, Unicast nodes are the default nodes. To create Multicast nodes the user must explicitly notify in the input OTcl script, right after creating a scheduler object, that all the nodes that will be created are multicast nodes. After specifying the node type, the user can also select a specific routing protocol other than using a default one.

- **Unicast**

  - *$ns* rtproto *type*
  - *type*: Static, Session, DV, cost, multi-path

- **Multicast**

  - *$ns* multicast (right after set *$ns* [new Scheduler])
  - *$ns* mrtproto *type*
  - *type*: CtrMcast, DM, ST, BST

For more information about routing, refer to the NS Manual located at http://www.isi.edu/nsnam/ns/ns-documentation.html. The documentation has chapters talk about unicast and multicast routing.

- **Link**

A link is another major compound object in NS. When a user creates a link using a duplex-link member function of a Simulator object, two simplex links in both directions are created as shown in Figure 8.

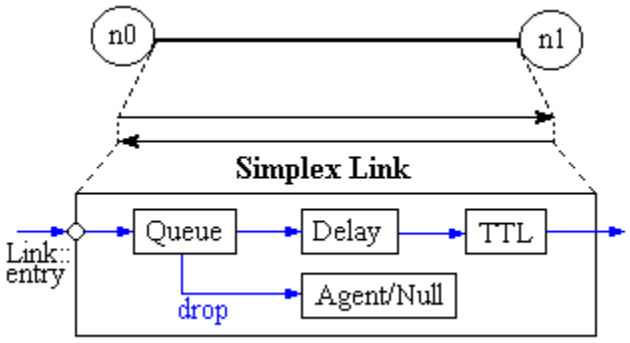

Figure 8. Link

One thing to note is that an output queue of a node is actually implemented as a part of simplex link object. Packets dequeued from a queue are passed to the Delay object that simulates the link delay, and packets dropped at a queue are sent to a Null Agent and are freed there. Finally, the TTL object calculates Time To Live parameters for each packet received and updates the TTL field of the packet.

- **Tracing**

  In NS, network activities are traced around simplex links. If the simulator is directed to trace network activities (specified using *$ns* trace-all *file* or *$ns* namtrace-all *file*), the links created after the command will have the following trace objects inserted as shown in Figure 9. Users can also specifically create a trace object of type *type* between the given *src* and *dst* nodes using the create-trace {*type file src dst*} command.

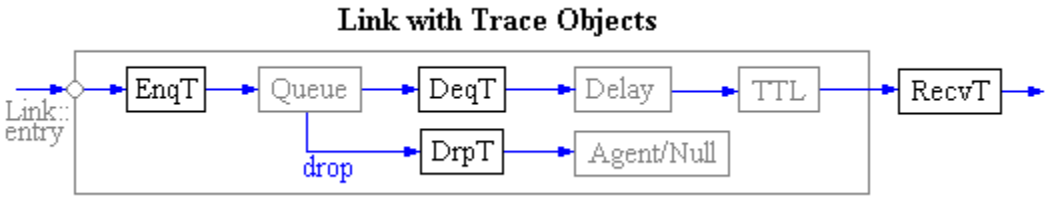

Figure 9. Inserting Trace Objects

When each inserted trace object (i.e. EnqT, DeqT, DrpT and RecvT) receives a packet, it writes to the specified trace file without consuming any simulation time, and passes the packet to the next network object. The trace format will be examined in the General Analysis Example section.

- **Queue Monitor**

  Basically, tracing objects are designed to record packet arrival time at which they are located. Although a user gets enough information from the trace, he or she might be interested in what is going on inside a specific output queue. For example, a user interested in RED queue behavior may want to measure the dynamics of average queue size and current queue size of a specific RED queue (i.e. need for queue monitoring).

Queue monitoring can be achieved using queue monitor objects and snoop queue objects as shown in Figure 10.

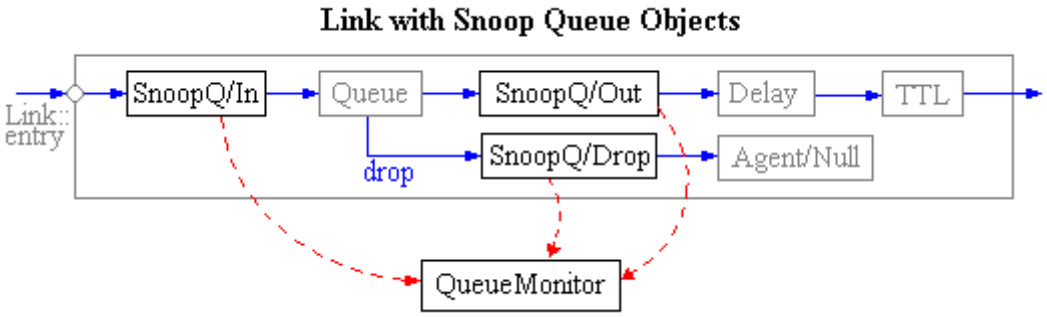

Figure 10. Monitoring Queue

When a packet arrives, a snoop queue object notifies the queue monitor object of this event. The queue monitor using this information monitors the queue. A RED queue monitoring example is shown in the RED Queue Monitor Example section. Note that snoop queue objects can be used in parallel with tracing objects even though it is not shown in the above figure.

- **Packet Flow Example**

Until now, the two most important network components (node and link) were examined. Figure 11 shows internals of an example simulation network setup and packet flow. The network consist of two nodes (n0 and n1) of which the network addresses are 0 and 1 respectively. A TCP agent attached to n0 using port 0 communicates with a TCP sink object attached to n1 port 0. Finally, an FTP application (or traffic source) is attached to the TCP agent, asking to send some amount of data.

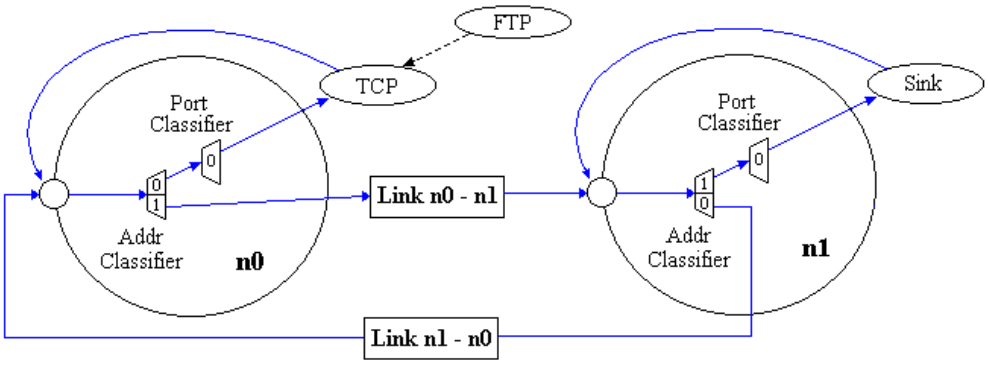

Figure 11. Packet Flow Example

Note that the above figure does not show the exact behavior of a FTP over TCP. It only shows the detailed internals of simulation network setup and a packet flow.