MONASH UNIVERSITY
DEPARTMENT OF ELECTRICAL & COMPUTER SYSTEMS ENGINEERING

Performance of Telecommunication Networks

Experiment II: Introductory Network Layer Protocols: Hot Potato Routing

Y. Ahmet Şekercioğlu

Introduction: Hot Potato Routing

We will now create a network of `tic` modules. Naturally, as soon as we allow our network to have more than two nodes, nodes will have to make routing decisions. That is, given a packet and its destination address, to which outgoing link should the packet be forwarded?

One of the simplest and most aesthetically pleasing routing algorithms is *hot potato* or *random routing* or *deflection routing*. When a packet arrives at a node, the node either keeps the packet (if the packet is addressed to the node) or sends it out on a randomly chosen link.

In hot potato routing, the nodes of a network have no buffers to store packets in before they are moved on to their final predetermined destination. In normal routing situations, when multiple packets contend for a single outgoing channel, packets that can not be buffered are dropped. But, in hot potato routing, to avoid dropping, each packet that is routed is constantly transferred until it reaches its final destination if an individual communication link is busy transmitting another packet (a communication link can not support more than one packet at a time). The packet is bounced around like a “hot potato”, inevitably sometimes moving further away from its destination because it has to keep moving through the network. This technique allows multiple packets to reach their destinations without being dropped.

Contrast this to “store and forward” routing where the network switches have temporary storage at intermediate locations to buffer packets whenever needed.

Hot potato routing has applications in optical networks where messages made from light can not be stored in any medium (http://webopedia.internet.com).

Exercise 1 - Our Very First Network

The exercise is accessible through the following Web page: http://www.omnetpp.org/doc/tictoc-tutorial/part3.html. Follow the steps given there and run your first simulation experiment. Afterwards answer the questions given in the following section.

Questions

1. How do we find the number of “out” gates of a module?

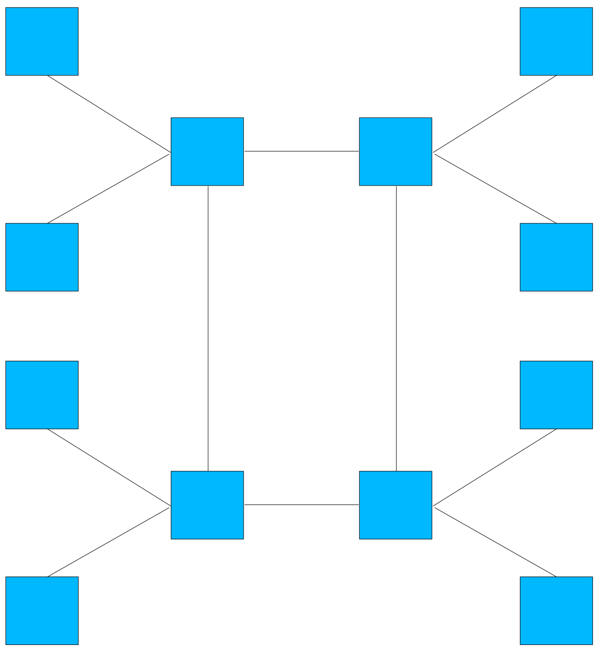

Figure 1: The 12-node `tictocnet`.

Exercise 2 - Learning About Module Gates

You probably have noticed the inefficiency of the hot potato routing. Often the packet keeps bouncing between two nodes for a while before it is sent to a different direction. This deficiency can be improved somewhat if nodes do not send the packet back to the sender. You can implement this functionality by finding out from which gate it has been received by the packet switch. You may find the following hints useful: `cMessage::arrivalGate()` and `cGate::index()` (see Section 5.1.3 of the OMNeT++ User Manual titled *Modelling packets* [Var06].

→ Show the completed model to your laboratory supervisor.

Questions

1. In the lectures, we have discussed that if a packet switch receives more than one packet to be forwarded to same destination, they have to be queued and one by one transmitted through the output port. In this exercise, we didn't need any queueing. Why?

Exercise 3 - Generating Periodic Messages

In this model, there is only one message travels at any given moment: a node only generates a new message when a message arrives at them. Locate in the module class the lines responsible for this limitation and change accordingly to make a node to generate messages periodically.

The interval (waiting period) between generation of two consecutive messages should be a module parameter returning exponentially distributed random numbers.

→ Show the completed model to your laboratory supervisor.

Exercise 4 - Creating a Larger Network

Create the 12-node `tictocnet` as shown in Figure 1, generate and run the simulation model executable. Which files should be modified to create this network topology?
→ Show the completed model to your laboratory supervisor.

References

[Var06] A. Varga. OMNeT++ Object-Oriented Discrete Event Simulation System User Manual. URL reference http://www.omnetpp.org/doc/manual/usman.html, 2006.