Department of Electrical Engineering
Sharif University of Technology

Creating Wireless Routing-Scheduling Agents with OPNET Simulator

Goals

1) To understand how to create simple agents with OPNET
2) To understand how to implement different layers in OPNET
3) To get familiar with routing and scheduling algorithms in wireless networks

Introduction

One of the well-known routing-scheduling algorithms in wireless networks is Optimum Routing-Scheduling Algorithm. This algorithm is optimum in the sense of maximizing the network throughput, so it is also known by the name Maximum Throughput Algorithm (MTA). In order to understand the MTA, consider the wireless network shown in Figure 1. It contains six nodes named $1,2,\dots,6$ that are connected via half-duplex wireless links. A main node named scheduler controls activities in the network. In a typical wireless networks, links may not be activated randomly, because it results in link interference and consequently an entity should coordinate link activation in the network. Link activation can be considered as a scheduling mechanism. In addition, it should be a routing procedure to route the packets of flows to their destination. But referring to link interference, only certain links and routes can be used in the same time, so routing and scheduling should be considered simultaneously in a wireless network. In other word, we need a cross layer design. In Figure 1, the scheduler is responsible for running a routing-scheduling algorithm.
In our desired topology, we assume one-hop interference. For example if link $2 \rightarrow 3$ is activated, the links $1 \rightarrow 2$, $2 \rightarrow 1$, $3 \rightarrow 2$, $2 \rightarrow 6, 6 \rightarrow 2$, $3 \rightarrow 4$ and $4 \rightarrow 3$ must be off. On the other hand, activation of the links $2 \rightarrow 3$ and $4 \rightarrow 5$ is a possible scenario. L is a set that contains all possible link activation scenarios:

$$L = \{\{2 \rightarrow 3, 4 \rightarrow 5\}, \{2 \rightarrow 3, 5 \rightarrow 4\}, \{2 \rightarrow 3, 5 \rightarrow 6\}, \dots\} \qquad \text{Equ(1)}$$

In the MTA, packets with different destinations are accumulated in separate queues. Q_i^j denotes the number of packets with destination node j, in node i. For each link $i \rightarrow k$, the MTA calculates the difference between separate queues $Q_i^j - Q_k^j$ and assigns its maximum value to the link $i \rightarrow k$ as a link weight w_{ik}:

$$w_{ik} = \max\{\max_j\{Q_i^j - Q_k^j\}, 0\} \qquad \text{Equ(2)}$$

Now, in the interval n, a set of links are chosen and activated such that no link interference is occurred and the following expression is maximized:

$$\max_{l \in L} \sum_{i \to k \in l} w_{ik}\mu_{ik} \qquad \text{Equ(3)}$$

Where l denotes members of the set L defined in Equ(2**)** and μ_{ik} is the capacity of the link $i \to k$.

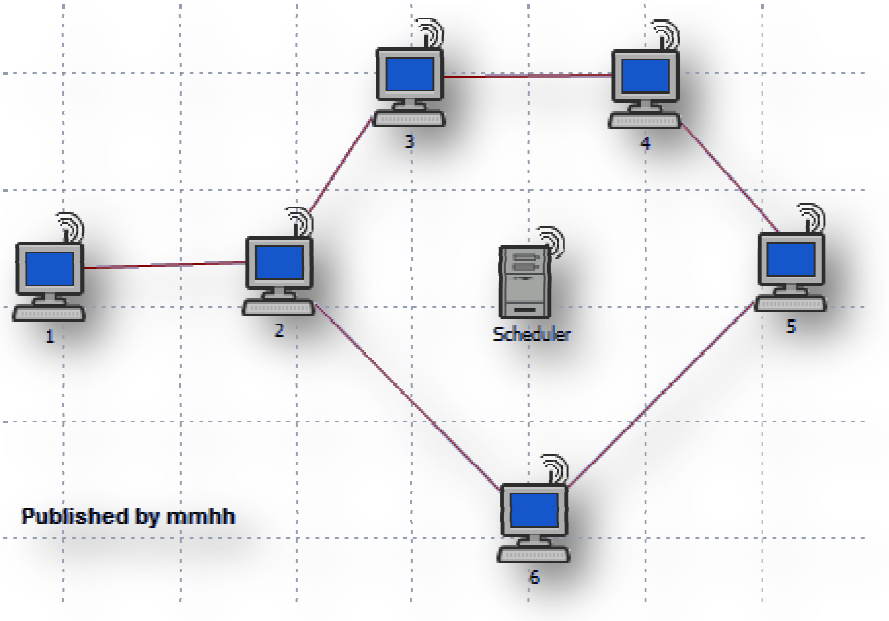

Figure 1: A Sample Wireless Network

In the desired network, the scheduler periodically polls all the nodes to know status of the queues and then activates the links determined according to the MTA. Although the MTA is originally a centralized algorithm and needs a main scheduler, but it can also be implemented by distributed techniques. However, distributed MTA cannot be optimum as the centralized one and its throughput may be degraded.

Here, we are going to implement the network of Figure 1 in OPNET simulator. We assume there are two flows from nodes 2 and 6 to nodes 5 and 4, respectively. We refer to the flow from node 2 to 5 as the first flow while the flow from 6 to 4 is named the second flow. Packets are generated according to Poisson times with parameter $R_{C_2} = 2R_{C_6}$ in application layers of nodes 2 and 6 and then fed to network layers to be routed based on the MTA. The scheduler periodically polls all the nodes each T_S, runs the MTA and then commands desired links to activate them. For simplicity, we assume:

1) There is no limitation on the size of queues.
2) All the links have the same capacity and no propagation delay.
3) All the packets have the same length.
4) Polling, running MTA and commanding is done instantly by the scheduler at the beginning of each interval T_S.
5) The polling rate is greater than the maximum of packet generation rates i.e. $1/T_S > \max\{R_{C_2}, R_{C_6}\}$.
6) The scheduler uses another frequency band so its interactions with the nodes have no interference with the links shown in Figure 1.
7) If there is more than one possible activation scenario, the scheduler chooses one of them randomly.
8) The received packets at the destination nodes 5 and 4 are dropped at a sink. Actually, they may only be used for measuring network parameters such as throughput, delay, … .

Node Model Definition

The Node Editor is used to create models of nodes. The node models are then used to create node instances within networks in the Project Editor. OPNET node models have a modular structure. You define a node by connecting various modules with packet streams and statistic wires. The connections between modules allow packets and status information to be exchanged between modules. Each module placed in a node serves a specific purpose, such as generating packets, queuing packets, processing packets, or transmitting and receiving packets.

In this project, you should design two node models. One of them is used for the scheduler and has only two layers that may negligently be named network and physical layers. This node model sends polling packets to the nodes and receives their answers through physical layer, runs the MTA in network layer and then sends command packets to activate determined links.

Another node model included in the rest of nodes in our topology may consist of three layers. The highest layer is application layer. This layer only generates packets based on Poisson times with the mentioned parameters R_{C_i} . $R_{C_i} = 0$ in all the nodes except nodes 2 and 6. A bit in the generated packets shows the destination of the sent packets. Note that in order to specify R_{C_i} as one the node's parameters you should first define it among the attributes and then promote it so that you can see it in higher hierarchy level. Refer to OPNET documentation for further help. By the way, T_S can be defined as the scheduler's parameters in the same way as R_{C_i} . The second level after the application level will be the routing level that has two main tasks. It should take data packets and put them in separate queues according to their destination. In addition, it should answer the scheduler' polling and send data packets based on the

scheduler's commands. Finally, the lowest layer is physical layer and is responsible for transmitting and receiving the data, polling and command packets.
Note that application and physical layers can easily be defined using OPNET predefined components but you should design appropriate state machines to be included in network layers of the scheduler and the other nodes. Each state machine may be equipped by desired and necessary states. Refer to OPNET documentation to know more.

The Packet Format Editor

By making use of this editor, it is possible to define the internal structure of a packet as a set of fields. A packet format contains one or more fields, represented in the editor as colored rectangular boxes. The size of the box is proportional to the number of bits specified as the field's size.
In this project, we may need three types of packets named polling, command and data packets. However, you can have your desired types, for example you may distinguish between the polling packets sent from the scheduler or the polling packets answered by the nodes. The poling packets may have fields like source name, destination name, queue size, link capacity, ... and command packets may have source name, destination name, command field, … while the data packets can consist of fields like source name, destination name, time stamp, hop-count, payload, … . Hop-count and time stamp fields are used for measuring delay and number of hops passed by a packet however OPNET has predefined component to measure this parameter and one can neglect these fields in the data packets. Remember that you should set the application layer of the nodes to create the data packet formatted using the packet format editor. Fortunately, we have a fixed and small topology in this project and we don't need to assign IDs to the nodes however adding them may be found useful. Although adding IDs can be done manually by adding a new field in packet format but OPNET has a feature named Object ID that removes the need for manual assigning of IDs and noticeably facilitates the project. For more information, refer to OPNET Kernel Procedures.

The Process Model Editor

To create process models which control the underlying functionality created in the Node Editor one can use the Process Editor. Processes are defined by finite state machines (FSMs) and are created with icons that represent transitions between states. Operations performed in the states are described in embedded C or C++ code blocks. You have access to a number of library functions that help you to do several network related operations. The package is referred to as proto-c. Refer to OPNET documentation for a list of functions you can use.

The Link Model Editor

In this project, we concern with wireless links however we virtually treat them as wired ones. One may use two types of links, one for connecting the nodes as shown in Figure 1 and other for communicating polling and commanding packets between the nodes and the scheduler. However, there are ways to communicate packets from the scheduler without adding new links. For example you can assign an array to each node and update it using proper OPNET kernel functions.

The link model editor enables you to create new types of link objects. Each new type of link can have different attribute interfaces and representations. Use this editor to design your favorite links.

Experiment Report

After developing your agents in the process editor and compiling them, make node models out of them to do simulation. You should define a number for statistics such as sent packets, received packets, hop-count, and delay to enable your network to collect the simulation results. Define a number of statistics for the nodes in the node editor. Then you should update those statistics using a group of proto-c functions starting with op-stat keyword. You are now able to see the results after running the simulation.
Now:

- ✓ Assign arbitrary and proper values to R_{C_i} and T_s. Run the simulation and record end-to-end delay, hop-count and queue sizes in the nodes and plot them.
- ✓ What are the main problems and weaknesses of the MTA? Refer to queue sizes or use OPNET ODB feature to trace packets to answer this question.
- ✓ Deactivate the links $6 \rightarrow 5$ and $5 \rightarrow 6$ using failure component at a certain time and run the simulation. What happens? Compare your results with the results of previous questions.
- ✓ Assume that the first flow has priority with respect to second one or in other words it has much QoS than the second flow. Offer a way for achieving this aim.
- ✓ **Back Pressure Algorithm** is another name used for the MTA. Noting the size of queues, explain why this name is used.

Project Report

Write a proper report using MS Word or Latex and include the results and discussions of your results. You should then pack this report with the project (and other necessary files to run your project as a standalone program) in a folder named as 'yourname_yourstudentnumber_Project', and then zip this folder and upload this one file to the courseware.
You should include the followings in your report:

- ✓ Explain all the steps in creating your agents. Use snapshots from the OPNET editors if required.
- ✓ Include the state machines and codes from OPNET process editors.
- ✓ Justify your results from the experimental part.

Added mark:
Each of the following parts is optional and you can get bonuses by solving them.

- ✓ Remember the no QoS scenario and assume that capacity of the links $2 \rightarrow 3$ and $3 \rightarrow 2$ is three times other links. Revise your project and simulate it. Compare the results with the previous parts.
- ✓ Here, we use a simple and fixed topology. Enhance your implementation to cover any arbitrary topology. You may found OPNET Object ID as a good feature. Remember that this is not a simple work however you can get a bonus by giving an intuitive explanation about this part.

- ✓ The MTA is inherently a centralized algorithm however it can approximately be implemented using distributed procedures. Find a distributed way and enhance your project.
  Hint: Refer to **Maximal Greedy Scheduling Approximation** as an example of distributed techniques.

Important note:
It is strongly offered to define two main structures in the scheduler's node model for collecting the required information. For example:

Table 1: Information of the Nodes

Node ID	Queue size (Flow 1)	Queue size (Flow 2)	In links	Out links
1				
2				
3				
4				
5				
6				

Table 2: Information of the links

Link ID	Begin node	End node	Capacity	Propagation Delay
$1 \rightarrow 2$	1	2	∞	0
$2 \rightarrow 1$	2	1	∞	0
$2 \rightarrow 3$				
$3 \rightarrow 2$				
$3 \rightarrow 4$				
$4 \rightarrow 3$				
$4 \rightarrow 5$				
$5 \rightarrow 4$				
$5 \rightarrow 6$				
$6 \rightarrow 5$				
$6 \rightarrow 2$				
$2 \rightarrow 6$				

These structures can facilitate finding possible scenarios of link activations and running the MTA. You may remove some structures' fields or expand them by adding new ones. Link structure can be used to generate link activation scenarios or equivalently the members of the set L in **Equ(1)** through a suitable code.