ENSC 835: HIGH-PERFORMANCE NETWORKS
CMPT 885: SPECIAL TOPICS: HIGH-PERFORMANCE NETWORKS

Comparative Analysis of Wireless Routing Algorithms in ns-2

Spring 2006

Demonstration Report

www.sfu.ca/~ekchen

Edward Chen (ekchen@sfu.ca)

Colin Ng (cnge@sfu.ca)

1 Project Overview

Before any analysis can be performed, there are several steps that have to be taken to accurately generate the desired results. This section provides an overview of the various applications used.

Figure 1 illustrates these steps in the form of a UML state diagram.

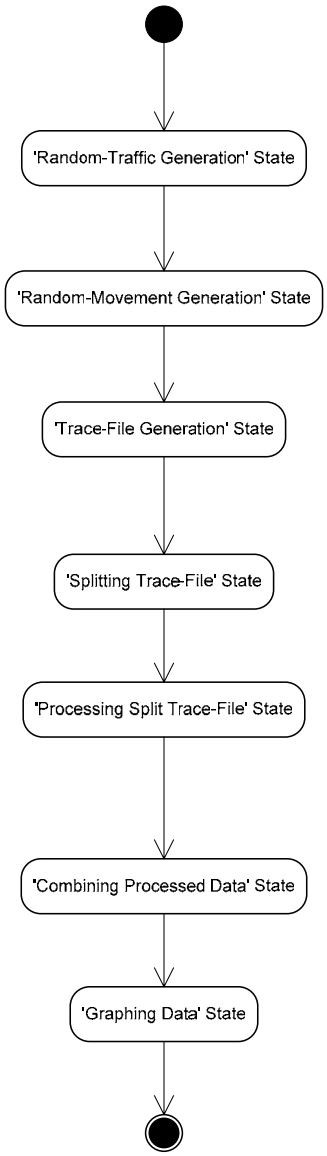

Figure 1 Steps Undertaken Before Analysis

The purpose of each state is summarized in Table 1, and will be explained in more detail in the following few sections.

State	Purpose of State
Trace-file generation	To generate trace-file
Splitting trace-file	To divide the trace-file into smaller pieces
Processing split trace-file	To process each individual trace-file piece
Combining processed data	To combine the processed data of each trace-file piece
Graphing data	To graph processed data

Table 1 Purpose of Each State

The following parameters are used for this demonstration. Because of time constraints, the total number of nodes used in this demonstration will be significantly less than that of the project.

Table 1 Demonstration Parameters

Parameter Name	Value
General Topology	
X-Boundary	1000 meters
Y-Boundary	1000 meters
Simulation Time	150 seconds
Routing Protocols	DSR
Node Movement	
Maximum Speed	5 m/s
Number of nodes	20
Pause time	1 sec
Traffic Generation	
Traffic Type	Constant Bit Rate (CBR)
Maximum Connections	10
Rate	5 kbps

To generate a random movement file, the following command is used.

```
./setdest [-n num_of_nodes] [-p pausetime] [-s maxspeed] [-t simtime] \
        [-x maxx] [-y maxy] > [outdir/movement-file]
```

To generate a random traffic file, the following command is used.

```
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc
        connections] [-rate rate]
```

1.1 'Trace-File Generation' State

The trace files are generated by executing a .tcl file with all the simulations parameters.

1.1.1 Performance Metrics

In this project, we are most interested in the following performance metrics:

Application Load:
The total number of combined application-related sent and application-related forwarded messages.

Received Load:
The total number of application-related received messages.

Dropped Load:
The total number of application-related dropped messages.

Routing Load:
The total number of combined routing-related sent and routing-related forwarded messages.

We will compare the three routing protocols with respect to the above four performance metrics. Also we will analyze each RP individually with respect to dropped and routed loads. Finally, we will compare network metrics such as end-to-end delay and throughput of the three routing protocols. All comparisons are made with respect to the project variables set forth in previous sections.

1.2 'Splitting Trace-File' State

When the trace-file-generation is complete, the data within the file would normally be processed directly. However, the time required to process a complete trace-file is extremely long (>24 hours per file). This is due to the logging of different data during the data-processing. Furthermore, the use of linked-lists, as opposed to arrays, also slows down the processing time.

As result of this slow processing time, it was decided that the more efficient path would be to divide the trace-file into smaller pieces, as show in Figure 2.

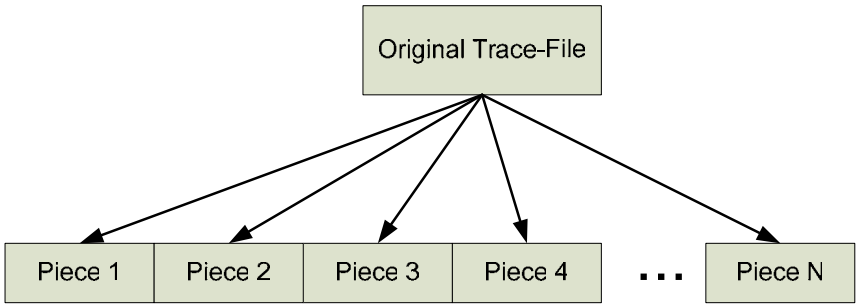

Figure 2 Splitting of trace-file into smaller pieces

The program written to perform the splitting of the trace-file is also written in Java. The UML activity diagram in Figure 3 shows the simple algorithm used.

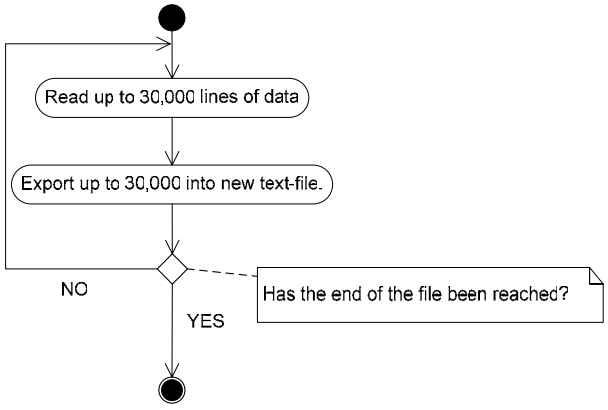

Figure 3 Splitting of trace-file into smaller pieces

1.3 'Processing Split Trace-File' State

1.3.1 Extraction of Data

The activity diagram for the program designed to do this is shown in Figure 4.

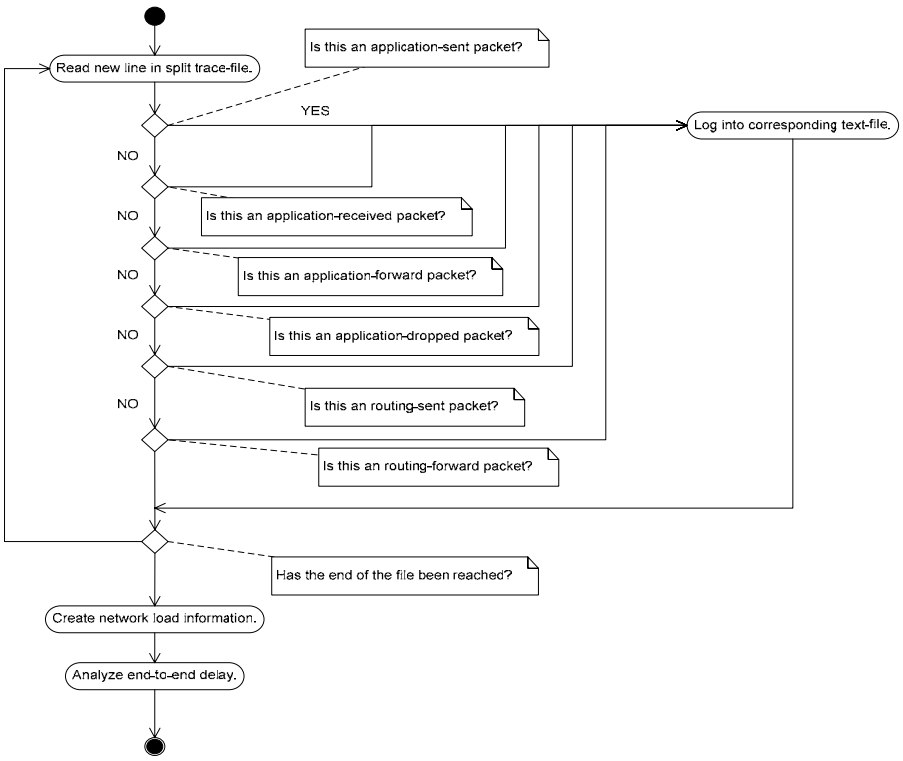

Figure 4 Processing of split trace-files

From the figure, it can be seen that three phases exist during the analysis – the data sorting phase, the network analysis phase, and the end-to-end delay analysis phase. In the first phase, the figure illustrates that six different types of packet are logged. The different checks that are performed during this phase split the data into two categories – routing-related data and application-related data. The routing-related information is used to update the routing tables, whereas the application-related data is the actual sending and forwarding of the application messages.

However, before any categorizing can be performed, it is important to first understand the logic flow of how the data is separated. There are two layers that the program will be tracking – the application layer and the routing layer. Distinguishing the difference between the two categories of data is done by analyzing the 18th field in the new wireless trace-file formats [1]. If the field value is "AGT", then it means that the layer being

accessed is the application layer. If the field value is “RTR”, then it means that the layer being accessed is the routing layer.

Next, it is important to understand that when a message is being sent and when a message is being received, it is the application layer that is being accessed, whereas messages that are forwarded and dropped are being in reality accessing the routing layer. The reason that the forwarded and dropped are accessing the routing layer, not the application layer, is because it is the router that performs the routing of application messages based on the routing table. Shown below are examples of application-related messages (where it can be seen that the 1st field defines the event such that sent messages are characterized by “s”, that received messages are characterized by “r”, that forwarded messages are characterized by “f”, and that dropped messages are characterized by “d”):

1. Example of sent message that is application-related

s -t 103.841413918 -Hs 27 -Hd -2 -Ni 27 -Nx 923.53 -Ny 437.20 -Nz 0.00 -Ne -1.000000 -Nl **AGT** -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 27.1 -Id 29.0 -It cbr -Il 512 -If 0 -Ii 7213 -Iv 32 -Pn cbr -Pi 311 -Pf 0 -Po 2

2. Example of received message that is application-related

r -t 103.838498889 -Hs 42 -Hd 42 -Ni 42 -Nx 143.79 -Ny 356.32 -Nz 0.00 -Ne -1.000000 -Nl **AGT** -Nw --- -Ma a2 -Md 2a -Ms 0 -Mt 800 -Is 41.1 -Id 42.1 -It cbr -Il 532 -If 0 -Ii 7079 -Iv 27 -Pn cbr -Pi 474 -Pf 4 -Po 3

3. Example of forwarded message that is application-related

f -t 103.831373999 -Hs 0 -Hd 42 -Ni 0 -Nx 266.77 -Ny 395.94 -Nz 0.00 -Ne -1.000000 -Nl **RTR** -Nw --- -Ma a2 -Md 0 -Ms 33 -Mt 800 -Is 41.1 -Id 42.1 -It cbr -Il 532 -If 0 -Ii 7148 -Iv 27 -Pn cbr -Pi 477 -Pf 3 -Po 3

4. Example of dropped message that is application-related

d -t 103.831373999 -Hs 0 -Hd 42 -Ni 0 -Nx 266.77 -Ny 395.94 -Nz 0.00 -Ne -1.000000 -Nl **RTR** -Nw --- -Ma a2 -Md 0 -Ms 33 -Mt 800 -Is 41.1 -Id 42.1 -It cbr -Il 532 -If 0 -Ii 7148 -Iv 27 -Pn cbr -Pi 477 -Pf 3 -Po 3

Routing-related messages are slightly different in that in addition to the 18th field still being checked to ensure that routing layer is accessed and to the 1st field still being checked for the event, the 34th field also has to be checked. If the routing protocol is AODV, this field will have a value of “AODV”. If the routing protocol is DSR, this field will have a value of “DSR”. DSDV however, poses a slightly different check and instead will have the value “message”. For routing-related messages, this project only inspects

sent and forwarded messages. The following is an example of a sent and a forwarded message for each of the three routing protocols:

1. AODV sent routing-related message

s -t 107.175544419 -Hs 59 -Hd -2 -Ni 59 -Nx 667.34 -Ny 298.73 -Nz 0.00 -Ne -1.000000 -Nl **RTR** -Nw --- -Ma 0 -Md ffffffff -Ms 27 -Mt 800 -Is 59.255 -Id -1.255 -It **AODV** -Il 52 -If 0 -Ii 0 -Iv 24 -P aodv -Pt 0x2 -Ph 6 -Pb 40 -Pd 20 -Pds 42 -Ps 18 -Pss 69 -Pc REQUEST

2. AODV forwarded routing-related message

f -t 107.175736691 -Hs 2 -Hd 18 -Ni 2 -Nx 496.24 -Ny 746.39 -Nz 0.00 -Ne -1.000000 -Nl **RTR** -Nw --- -Ma a2 -Md 2 -Ms 22 -Mt 800 -Is 23.255 -Id 18.255 -It **AODV** -Il 44 -If 0 -Ii 0 -Iv 26 -P aodv -Pt 0x4 -Ph 8 -Pd 20 -Pds 43 -Pl 57 -Pc REPLY

3. DSR sent routing-related message

s -t 16.414914946 -Hs 12 -Hd 42 -Ni 12 -Nx 348.87 -Ny 218.47 -Nz 0.00 -Ne -1.000000 -Nl **RTR** -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 12.255 -Id 32.255 -It **DSR** -Il 84 -If 0 -Ii 300 -Iv 255 -P dsr -Ph 4 -Pq 0 -Ps 6 -Pp 1 -Pn 6 -Pl 12 -Pe 32->34 -Pw 0 -Pm 0 -Pc 0 -Pb 0->0

4. DSR forwarded routing-related message

f -t 16.404069831 -Hs 13 -Hd -1 -Ni 13 -Nx 128.61 -Ny 406.71 -Nz 0.00 -Ne -1.000000 -Nl **RTR** -Nw --- -Ma 0 -Md ffffffff -Ms 33 -Mt 800 -Is 32.255 -Id 34.255 -It **DSR** -Il 68 -If 0 -Ii 296 -Iv 32 -P dsr -Ph 3 -Pq 1 -Ps 6 -Pp 0 -Pn 6 -Pl 0 -Pe 0->0 -Pw 1 -Pm 1 -Pc 0 -Pb 46->34

5. DSDV sent routing-related message

s -t 124.120999159 -Hs 49 -Hd -1 -Ni 49 -Nx 190.53 -Ny 827.07 -Nz 0.00 -Ne -1.000000 -Nl **RTR** -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 49.255 -Id -1.255 -It **message** -Il 320 -If 0 -Ii 12695 -Iv 32

6. DSDV forwarded routing-related message

No forwarded routing-related messages are tracked because the DSDV implementation in ns-2 does not perform this. Instead, the implementation receives the message and then resends the message. So, the forwarded message is actually treated as a new sent message.

All these categories are created because these are the different types of data that the project will be using as the basis for evaluating the performances of the different routing protocols. In the second phase, an evaluation of the logged data is done. This will create several new text-files, where in them exists different network characteristics. The final phase completes the analysis by evaluating the end-to-end delay of the network.

1.3.2 Calculation of Performance Metrics

As mentioned in Section 1.1.1, the four performance metrics being calculated are:

- Application Load
- Application Received Load
- Application Dropped Load
- Routing Load.

In addition to these four network characteristics, the end-to-end delay of the entire network and the network throughput will also be analyzed in this project.

1.3.2.1 Calculation of Application Load

Application load is defined as the addition of the application messages sent and the application messages forwarded. For instance, if there are 1101 messages sent and 1000 messages forwarded, the application load would be 2101.

1.3.2.2 Calculation of Received Load and Dropped Load

Each time a message is received, a counter that keeps track of the total number of messages received is incremented. Similarly, each time a packet is dropped, a counter that keeps track of the total number of dropped messages is incremented.

1.3.2.3 Calculation of Routing Load

Routing load is defined as the addition of the routing messages sent and the routing messages forwarded. For instance, if there are 2101 messages sent and 1000 messages forwarded, the application load would be 3101.

1.3.2.4 Calculation of End-To-End Delay

Each time a message is sent, the time that it was sent is stored in memory. When the message has been received by a node, that sent time is accessed and compared with the received time. The difference between the two times is the end-to-end delay. For instance, if the sent time is 3.01 seconds, and if the received time is 5.01 seconds, the end-to-end delay is 2 seconds.

1.3.2.5 Calculation of Network Throughput

The network throughput is calculated towards the end of the data processing. Defined as the number of packets sent per second, the throughput is calculated by dividing the application sent load by the total simulation time.

1.4 'Combining Processed Data' State

When the evaluations have been completed, the result is a set of text-files containing the network characteristics for each split piece of the original trace file. This set however, is not useful to the user yet because the information placed into each individual piece is only a portion of the total network information. So, a Java program was written to combine the individual results. Figure 5 illustrates the logic flow behind the combination.

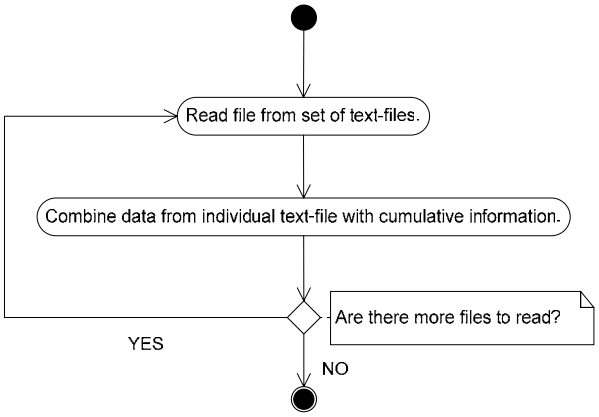

Figure 5 Combining processed pieces

1.5 'Graphing Data' State

Graphs used for analysis were produced using Matlab based on the data points that were extracted from the complete processed data. Matlab was used because Microsoft Excel would not allow more than 32,000 data points to be plotted; this project produced graphs where very often there were more than 32,000 data points that had to be plotted.