Implement a routing agent with manually configured routing table (Static Routing)

[Back to *Network Simulator 2 for Wireless*]

Outline

1. Agent and Connector

Connector is one of the most basic classes in ns-2. It has a "target_" object , handles "recv", "send" and "drop" activities. Agent is a child class of Connector.
Basically, a routing agent is going to override "recv" function of agent.
The default behavior of "recv" in Agent is to calling the *app_* 's "recv" function.
However, as routing agent is usually independent of the application, it has to override the virtual "recv" function.

We implement the new routing agent by modifying codes from DSDV source code in ns-2. As DSDV in ns-2 defines two additonal auxillary classes, the static routing agent also defines two new classes.

```
class FixRTable;   // routing table
class fixrt_ent;  // routing table entry
```

2. Routing Table Class.

FixRTable is a class that store routing table entries. Each entry should have at least <dst, next-hop and metric>. Often in DSDV or AODV, it has sequence numbers to determine if it is a newer routing message or not. For fixed routing, this is unnecessary. To use this new agent, change opt(rt) from AODV or DSDV to FIXRT in your wireless simulation script.

In almost every routing protocol design, a packet queue is used to store those packets with valid destination but without valid routes yet. This is unnecessary for this static routing design. Thus, the router will no longer need to have queues for each outgoing ports, we remove IP layer queues in "PacketQueue" member variable in class "fixrt_ent".

3. Routing Agent Class.

The most important task for a routing agent is to detect link metrics and refresh

routing table with special routing signaling messages such as periodic advertisements. However, in static routing, there is no such a burden. Only two things need to be addressed:
1) Start-up function, called in the command "start-fixrt". In this function, we call *makeRoutingtable* function. routing information will be "fscanf" from a file and adding to the tables. Here, we are going to use "fscanf" to read three parameters. "int32" "int32 ""uint16", to represent <dst, hop, metric>. The filename must be specified. Different nodes read different sections of the routing table file. The special format of this file is:

<src><dst> <next-hop><number of hops>
0 3 1 3
.....

So, only those entries where 'src' addr is equal to its addresss will be read by a node. The auto-generation of routing table could be easily implemented by an algorithm from the topology file.

2) Packets Forwarding functions: according to the routing table, the agent needs to handle both "broadcast" and "unicast" packets. "Forwarding" means determining the next-hop of a unicast packet. Once this field of the header is set, it is done. In FIXRT, broadcast occurres only locally.

The agent's interface to MAC componet is a little complex:
1) Message interface: packets going to mac with next-hop set, ready for layer 2 transmission.
2) Function interface: once link of loss is detected, the MAC will "callback" (inform) routing agent to do correspondingly. In this Fix Routing agent, this would be useless. The agent will do nothing for "mac_callback" function in this static routing design.

4. Interface (Linkage) to Otcl

When a wireless node is created, its default routing agent is specified in the arguments, such as opt(rt). In /tcl/lib/ns-lib.tcl. the **"create-wireless-node"** procedure will check the routing protocol type. Here we can add a new type as "FIXRT" in the "switch" block. Also add "set ragent [$self create-fixrt-agent $node]". For example,

```
switch -exact $routingAgent_ {
           DSDV {
                   set ragent [$self create-dsdv-agent $node]
           }
           FIXRT {
                   set ragent [$self create-fixrt-agent $node]
           }
  ...
  }
```

the "**create-fixrt-agent**" procedure is also need to be defined. In this procedure, a "$self at 0.0 "$ragent start-fixrt" " will be given to the routing agent. Therefore, we

have to initialize routing table in the start-fixrt command.

```
Simulator instproc create-fixrt-agent { node } {
        # Create a fix-path routing agent for this node
        set ragent [new Agent/FixRT]
        # Setup address (supports hier-addr) for dsdv agent
        # and mobilenode
        set addr [$node node-addr]
        $ragent addr $addr
        $ragent node $node
        if [Simulator set mobile_ip_] {
                $ragent port-dmux [$node demux]
        }
        $node addr $addr
        $node set ragent_ $ragent
        $self at 0.0 "$ragent start-fixrt"    ;# start updates
        return $ragent
}
```

After this, all packets coming will be handled by the "recv" function of the agent, and the agent will just route packets according to the table.

To use this new agent, change opt(rt) from DSDV/AODV to FIXRT in your wireless simulation script.

5. Benefits of Using Fix-Route Routing

When FIXRT is used as a routing protocol instead of DSDV, AODV, DSR, it is a more efficient routing protocol because the route is pre-calculated in an optimal way (e.g. shortest path). Thus, it stands as an upper bound of performance in all routing protocols, if nodes are not moving. When nodes are mobile, as there are no adaptive changes for routing table, the performance will be sub-optimal.

The other big benefits is that routing messages are suppressed, and link loss are also ignored. So, it provides a basic and fair platform for comparing MAC protocol performance with the "**ALWAYS SAME**" route.!!!
This is important because routing protocol depends on MAC to propagate routing messages, If MAC is modified, the routing path could also be changed and the performance comparison is no longer accurate!

APPENDIX

Routing Table for 4*4 grid

0----1-----2-----3
| | | |
4----5-----6-----7
| | | |
8----9-----10---11
| | | |

12--13---14----15

0 1 1 1
0 2 1 2
0 3 1 3
0 4 1 2
0 5 1 3
0 6 1 4
0 7 1 3
0 8 1 3
0 9 1 4
0 10 1 4
0 11 1 5
0 12 1 5
0 13 1 6
0 14 1 4
1 0 0 1
1 2 2 1
1 3 2 2
1 4 4 1
1 5 4 2
1 6 4 3
1 7 4 2
1 8 4 2
1 9 4 3
1 10 4 3
1 11 4 4
1 12 4 4
1 13 4 5
1 14 2 3
2 0 1 2
2 1 1 1
2 3 3 1
2 4 5 2
2 5 5 1
2 6 5 2
2 7 5 3
2 8 5 2
2 9 5 2
2 10 5 4
2 11 5 3
2 12 5 3
2 13 5 4
2 14 3 2
3 0 2 3
3 1 2 2
3 2 2 1
3 4 6 3

3 5 6 2
3 6 6 1
3 7 6 4
3 8 6 3
3 9 6 2
3 10 6 5
3 11 6 3
3 12 6 2
3 13 6 4
3 14 14 1
4 0 1 2
4 1 1 1
4 2 5 2
4 3 5 3
4 5 5 1
4 6 5 2
4 7 7 1
4 8 8 1
4 9 8 2
4 10 7 2
4 11 8 3
4 12 8 3
4 13 8 4
4 14 5 4
5 0 4 3
5 1 4 2
5 2 2 1
5 3 6 2
5 4 4 1
5 6 6 1
5 7 8 2
5 8 8 1
5 9 9 1
5 10 8 3
5 11 9 2
5 12 9 2
5 13 9 3
5 14 6 3
6 0 5 4
6 1 5 3
6 2 5 2
6 3 3 1
6 4 5 2
6 5 5 1
6 7 9 3
6 8 9 2
6 9 9 1
6 10 9 4

6 11 12 2
6 12 12 1
6 13 12 3
6 14 3 2
7 0 4 3
7 1 4 2
7 2 8 3
7 3 8 4
7 4 4 1
7 5 8 2
7 6 8 3
7 8 8 1
7 9 8 2
7 10 10 1
7 11 8 3
7 12 8 3
7 13 8 4
7 14 8 5
8 0 4 3
8 1 4 2
8 2 5 2
8 3 9 3
8 4 4 1
8 5 5 1
8 6 9 2
8 7 7 1
8 9 9 1
8 10 7 2
8 11 9 2
8 12 9 2
8 13 9 3
8 14 9 4
9 0 8 4
9 1 8 3
9 2 5 2
9 3 6 2
9 4 8 2
9 5 5 1
9 6 6 1
9 7 8 2
9 8 8 1
9 10 8 3
9 11 11 1
9 12 12 1
9 13 11 2
9 14 6 3
10 0 7 4
10 1 7 3

10 2 7 4
10 3 7 5
10 4 7 2
10 5 7 3
10 6 7 4
10 7 7 1
10 8 7 2
10 9 7 3
10 11 7 4
10 12 7 4
10 13 7 5
10 14 7 6
11 0 9 5
11 1 9 4
11 2 9 3
11 3 12 3
11 4 9 3
11 5 9 2
11 6 12 2
11 7 9 3
11 8 9 2
11 9 9 1
11 10 9 4
11 12 12 1
11 13 13 1
11 14 12 4
12 0 9 5
12 1 9 4
12 2 9 3
12 3 6 2
12 4 9 3
12 5 9 2
12 6 6 1
12 7 9 3
12 8 9 2
12 9 9 1
12 10 9 4
12 11 11 1
12 13 11 2
12 14 6 3
13 0 11 6
13 1 11 5
13 2 11 4
13 3 11 4
13 4 11 4
13 5 11 3
13 6 11 3
13 7 11 4

13 8 11 3
13 9 11 2
13 10 11 5
13 11 11 1
13 12 11 2
13 14 11 5
14 0 3 4
14 1 3 3
14 2 3 2
14 3 3 1
14 4 3 4
14 5 3 3
14 6 3 2
14 7 3 5
14 8 3 4
14 9 3 3
14 10 3 6
14 11 3 4
14 12 3 3
14 13 3 5

Sourcecode:

There are two classes implemented. So, there are totally 4 files;

- Fix Routing Agent Class
  - fixrt.cc
  - fixrt.h
- Table Class
  - fixtb.cc
  - fixtb.h

After add those 4 files, you also need to change ns-lib.tcl file and Makefile before you make ns-2.

There are two matlab files which are useful to generating the static routing table for complex topologies. The required arguments for the matlab function is the metric array (weighted adjacency array) and source, destination vectors.

- shortest_path_routing.m
- dijkstra_wu.m

Note that the "node" array in ns-2 begins from index 0. While in matlab, it begins from index 1.