Simulation of the Global Routing Protocol based on NS-3

Keng Ye
School of Information and Communication Engineering
Beijing Information Science and Technology University
Beijing, China
e-mail: yekeng@163.com

Jinhe Zhou
School of Information and Communication Engineering
Beijing Information Science and Technology University
Beijing, China
e-mail: zhoujinhe@bistu.edu.cn

***Abstract*—NS-3 (Network Simulator version 3) could construct a network environment to simulate protocols in computer network. The purpose of this paper is implementing global routing by using the NS-3 with modified OSPF source code. The processing of simulation had been described with C++ code. Furthermore, the result of the network simulation had been analyzed with different software tools such as gawk, gunplot. In this paper we focused on simulating the global routing of Internet by NS-3 and analyzing the related theories about the routing protocol, at last, this paper was analyzing the average delay under different rates.**

Keywords- routing protocol; NS-3; OSPF; simulation

I. INTRODUCTION

NS-3 not only had abandoned shortcomings of the current mainstream network simulation software such as OPNET and NS-2, but also integrated the advantages of them. NS-3 had excellence including the following aspects: (1) documenting kernel of the standard components; (2) using scripting language such as C + + or Python; (3) applying to real system (network interface, device, driver interface) perfectly; (4) software integration; (5) virtualization and test-bed integration; (6) excellent documented property system; (7) updating the model in real time.

NS-3 is a kind of unique simulator which includes integrity, openness, scalability and etc, and these feathers make it superior to most of the existing mainstream network simulator software. NS-3 has extremely powerful function to simulate a variety of network, protocols at all levels.

NS-3 is different from the NS-2, although NS-3 is still using the C++ language to realize the simulation node. It hadn't supported on NS-2 API and obsoleted Otcl language to control the processing of simulation. The framework of NS-3, whose simulation process can be described by pure C++ code, is much clearer than other software [1].

II. THE PRINCIPLE OF GLOBAL ROUTING PROTOCOL

In NS-3, a C++ object builds an OSPF (Open Shortest Path First) routing database of information about the network topology [2], and executes a Dijkstra SPF (Shortest Path First) algorithm on the topology for each node, and stores the computed routes in each node's layer 3 forwarding table by making use of the routing API(Application Program Interface). The format of the data exported harmonizes with the OSPFv2 standard. In particular, the information is exported in the form of **ns3::GlobalLSA** objects that semantically conform to the LSA (Link State Advertisements) of OSPF. By utilizing a standard data format for informing topology, existing OSPF route computation code can be reused, and that is what can be done by **ns3::GlobalRouteManager** objects**.**

OSPF is one kind of IGP (Interior Gateway Protocol), which is used for making routing decision in a single AS (Autonomous System). OSPF is a kind of link-state routing protocol, while RIP is a kind of distance vector routing protocol. In the other words, link is called router interface, so OSPF is also known as the interface state routing protocol. The OSPF is establishing the link state database, generating the shortest path tree through the network interface state of router notice, each OSPF router using the shortest path to construct the routing table [3].

The routing algorithm is the core of the OSPF routing protocol. The SPF algorithm is also called Dijkstra algorithm as shown in Fig. 1, because of the shortest path first algorithm SPF is invented by Dijkstra. The SPF algorithm utilize each router as root node to calculate the distance from root node to each destination router, according to a unified database each router will be worked out the topological structure in routing domain, which is similar to a tree called SPT(shortest path tree)in the SPF algorithm. In the OSPF routing protocol, the trunk length of SPT is the distance from the root router to other routers. Smaller Cost means that the OSPF distance between source and destination is shorter.

S represents the set of nodes which have not found the shortest path to the root node.

R [i] represents the node which is located in front the node i on the path from the specified source node to node i.

D [i] represent the shortest distance from the specified source to the node i.

Initialization of the algorithm:

The set **S** is initialized as a set which include all nodes but the source node.

D[i]: If there is a link from the source node to node v, **D (v)** is the Cost of the link; otherwise **D (v)** is infinity.

R[i]: If there is a link from source node to node v, **R (v)** is source node; otherwise **R (v)** =0

```
while(S is not empty)
      { choose a node u from S to minimum
D[u];
       if(D[u] is infinite)
         {error! no path exists,exit;}
            delete u from S;
            pair (u,v) is each node v of side;
            if(S still includes v)
            {C=D[u]+weight(u,v);
             if (C<D[v])  /*v find a shorter
path*/
             {
             /*replace shortest path and the
cost */
             R[v]= u;
             D[v]=C; }
                        }
             }
```

Figure 1. Pseudo code for Dijstra algorthm

When the router is initializing or the network structure is changing (for example, increasing or decreasing in the router, link state changing), the router will generate LSA, which contains all of the routers connected to the link, and calculate the shortest path.

All routers exchange data of link state with each other through a Flooding, which is that the routers sent LSA to all adjacent OSPF routers. Basing on which the routers received, they have been updating their database and forwarding link state information to the adjacent routers until a stable process

When the network is re-stabilized, the convergences of OSPF routing protocol have been completed. All routers will be calculated by the respective link state database of information to generate routing table, which contains the router to another reachable one and the destination to the next router.

III. THE SIMULATION PROCESS OF GLOBAL ROUTING

In this section, we'll introduce some important terms that are realizing specific function in ns-3.

A. Node

In NS-3 the basic abstraction of network device is called the **Node**. The class Node provides methods for managing the functions of network devices in simulation. You should consider a **Node** as a computer or a router to which you will add functionality by yourself [1]. One adds things like topology, channel, protocol stacks and peripheral devices with their associated drivers to activate the **Node** to do practical function.

Class **NodeContainer** provides a very convenient way to create, manage, and access nodes, set as follows,

NodeContainer c;
c.Create (7);

B. Topology

Class **NodeContainer** can connect any two nodes, set as follows,

NodeContainer n0n4 = NodeContainer(c.Get(0), c.Get (4));

C. Channel

The class **Channel** provides methods for constructing communication subnetwork objects and connecting nodes to each other. Channels may also be specialized by us in the object- oriented programming method. The specialized **Channel** can simulate things as complicated as a wire, a large Ethernet switch, or three-dimensional space full of obstructions in the case of wireless networks [1].

We will utilize specialized versions of the **Channel** called **CsmaChannel**, **PointToPointChannel** and **WifiChannel** in NS-3. This paper chooses **PointToPointHelper** to simulate, set as follows,

PointToPointHelper p2p;

D. NetDevice

In ns-3 the abstraction of network device covers both the software driver and the simulated hardware. A network device is "installed" in a **Node** so as to activate the Node to communicate with other Nodes in the simulation via Channels. Just as in a real computer, a Node may be connected to more than one **Channel** via multiple **NetDevices**. The abstraction of network device is represented in C++ by the class **NetDevice**. The class **NetDevice** provides methods for managing connections to Node and Channel objects; and may be specialized by us in the object-oriented programming. We will utilize the several specialized versions of the **NetDevice** called **CsmaNetDevice**, **PointToPointNetDevice**, and **WifiNetDevice** in NS-3[1]. This paper chooses **PointToPointNetDevice** to simulate, set as follows,

p2p.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
NetDeviceContainer d0d4 = p2p.Install (n0n4);

E. Protocol Stack

Until now, nodes, devices and channels are created, we will use class **InternetStack** to add stack, set as follows,

InternetStackHelper internet;
internet.Install (c);

F. Assignning Ipv4 Address

This paper chooses Class Ipv4AddressHelper assign IP addresses to the device interfaces.

NS_LOG_INFO ("Assign IP Addresses.");
Ipv4AddressHelper ipv4;
ipv4.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer i0i4=ipv4.Assign (d0d4);

G. Setting OSPF Cost Metric

i0i4.SetMetric (0, sampleMetric04);
i0i4.SetMetric (1, sampleMetric04);

H. Generating Routing Table

NS_LOG_INFO is recording information on the progress of the program. This paper chooses global routing protocol which is introduced in section 2.

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

I. Sending Traffic And Setting Data Rate , The Size Of Packet

```
NS_LOG_INFO ("Create Applications.");
uint16_t port = 80;   // Discard port (RFC 863)
OnOffHelper onoff ("ns3::TcpSocketFactory",
InetSocketAddress (i5i6.GetAddress (1), port));
onoff.SetAttribute("OnTime",
RandomVariableValue (ConstantVariable (1)));
onoff.SetAttribute("OffTime",
RandomVariableValue (ConstantVariable (0)));
onoff.SetAttribute("DataRate",StringValue("5kbps")
);
onoff.SetAttribute ("PacketSize", UintegerValue (50));
```

J. Generating And Stopping Traffic

```
ApplicationContainer apps = onoff.Install (c.Get (0));
apps.Start (Seconds (1.0));
apps.Stop (Seconds (100.0));
```

K. Class Packetsink Receive The Traffic Of

```
PacketSinkHelper sink ("ns3::TcpSocketFactory",
                Address (InetSocketAddress
(Ipv4Address::GetAny (), port)));
apps = sink.Install (c.Get (6));
apps.Start (Seconds (1.0));
apps.Stop (Seconds (100.0));
NS_LOG_INFO ("Configure Tracing.");
```

L. Generating Trace File To Record Simulation Information

```
AsciiTraceHelper ascii;
Ptr<OutputStreamWrapper> stream =
ascii.CreateFileStream ("ospf.tr");
p2p.EnableAsciiAll (stream);
internet.EnableAsciiIpv4All (stream);
```

M. Generating Routing File To Record Route Information

```
Ipv4GlobalRoutingHelper g;
Ptr<OutputStreamWrapper>routingStream=
Create<OutputStreamWrapper>("ospf.routes",
std::ios::out);
g.PrintRoutingTableAllAt(Seconds(12),
routingStream);
```

N. Executing Simulator

```
NS_LOG_INFO ("Run Simulation.");
Simulator::Run ();
Simulator::Destroy ();
NS_LOG_INFO ("Done.");
```

IV. SIMULATION RESULT

Simulation results verify that the shortest path tree conforms to Dijkstra algorithm in OSPF protocol and count package delay to check network performance.

A. Generating Shortest Path

According to the topology in Fig. 2, in the 1st second executes a Dijkstra SPF algorithm on the topology for each node, finds the shortest path between node 0 and node 3, as shown in Fig. 3 the shortest path is node 0->node 4-> node 1-> node 2-> node 5-> node 6-> node 3, rather than node 0->node 4-> node 1-> node 2-> node 3 which include the least node.

In 2nd second, disconnecting the link node 1 and node 2, node 0 executes a Dijkstra SPF algorithm on the topology for each node, finds the shortest path between node 0 and node 3, as shown in Fig.4 at the moment, the shortest path is node 0->node 4-> node 1-> node 5-> node 6 -> node 3.

In 4th second, reconnecting the link node 1 and node 2, node 0 executes a Dijkstra SPF algorithm on the topology for each node, finds the shortest path between node 0 and node 3, the shortest path is node 0->node 4-> node 1-> node 2-> node 5-> node 6-> node 3.

In 6th second, disconnecting the link node 5 and node 6,node 0 executes a Dijkstra SPF algorithm on the topology for each node, finds the shortest path between node 0 and node 3, at the moment, the shortest path is node 0->node 4-> node 1-> node 2-> node 3.

In 8th second, disconnecting the link node 2 and node 3,node 0 executes a Dijkstra SPF algorithm on the topology for each node, finds the shortest path between node 0 and node 3, at the moment, the shortest path is node 0->node 4-> node 1-> node 5-> node 2 -> node 3.

In 12th second, reconnecting the link node 1 and node 2, node 0 executes a Dijkstra SPF algorithm on the topology for each node, finds the shortest path between node 0 and node 3, at the moment, the shortest path is node 0->node 4-> node 1-> node 5-> node 6 -> node 3.

In 14th second, reconnecting the link node 1 and node 2, node 0 executes a Dijkstra SPF algorithm on the topology for each node, finds the shortest path between node 0 and node 3, at the moment, the shortest path is node 0->node 4-> node 1-> node 2-> node 5-> node 6-> node 3.

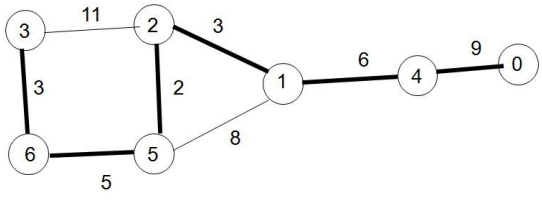

Figure 2. topology with cost in simulation

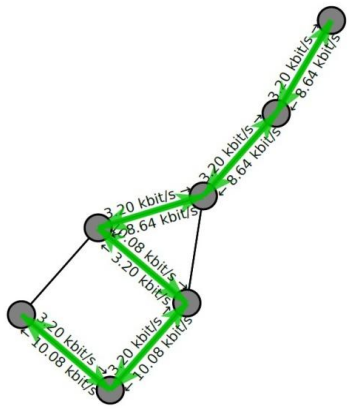

Figure 3. shortest path in 1st second

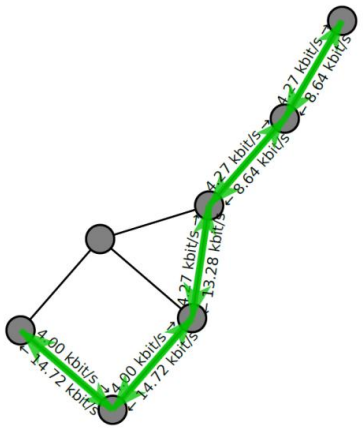

Figure 4. shortest path in 2nd second

B. *Delay*

In Fig. 5 are shown the simulation results of delay for the communication between node 0 and node 3. Under different shortest path, delay is diverse. The network delay is very close to the case when the packets take the same path. The path of least delay is not necessarily the shortest path.

The average delay is the time of all data packets arrived at destination node from the source node. The characterization means the current status of the network. As shown in Fig. 6,

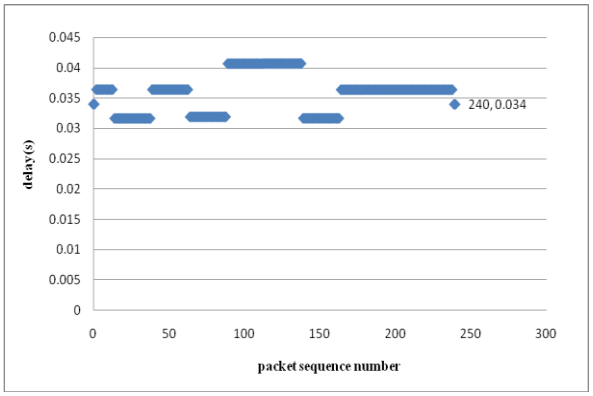

Figure 5. the packet delay

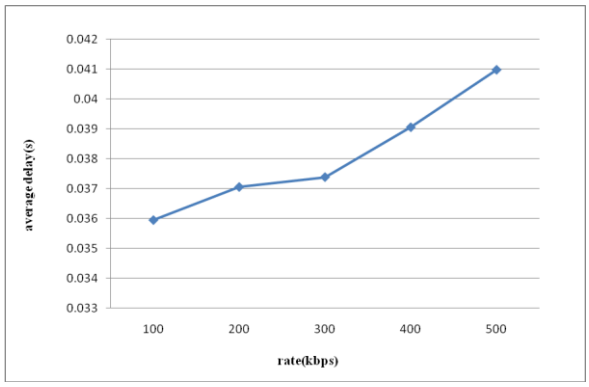

Figure 6. the average delay under different rate

V. CONCLUSION

Summary, this paper describes an effective simulation method to realize global routing protocols, evaluate network performance, and carry out a detailed analysis of delay and shortest path tree. The conclusion has certain reference value.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China (61271198) and Beijing Natural Science Foundation (4131003).

REFERENCES

[1] NS-3 Tutorial. http://www.nsnam.org/, 12 Dev 2012

[2] NS-3 project NS-3 Doxygen. http://www.nsnam.org/,12 Dev 2012

[3] Andrew S. Tanenbaum. Computer Network Fourth Edition. Beijing: Tsinghua University Press.pp.350-366,454-459(2008)

[4] WANG Jianqiang,LI Shiwei,ZENG Junwei,DOU Yingying.(2011)Simulation Research of VANETs Routing Protocols Performance Based on NS -3Microcomputer Applications Vol. 32 No. 11.

[5] NS-3 project NS-3 Reference Manual. http://www.nsnam.org/,12 Dev 2012

[6] Timo Bingmann.(2009). Accuracy Enhancements of the 802.11 Model and EDCA QoS Extensions in ns-3. Diploma Thesis at the Institute of Telematics

[7] Learmonth, G. Holliday, J. (2011) NS3 simulation and analysis of MCCA: Multihop Clear Channel Assessment in 802.11 DCF .Consumer Communications and Networking Conference (CCNC). IEEE

[8] Henderson T R,Floyd S,Riley G F(2006) NS-3 Project Goals. Proceedings of the Workshop of Network Simulation. Pisa, Italy.

[9] Vincent.S,Montavont.J(2008).Implementation of an IPv6 Stack for NS-3. Proceedings of the Workshop of Network Simulation.AthensGreece:[s. n.].