IMPLEMENTACIJA MODIFIKOVANOG OSPF PROTOKOL U MREŽNI SIMULATOR
IMPLEMENTATION OF MODIFIED OSPF PROTOCOL IN NETWORK SIMULATOR

Biljana Citkuseva Dimitrovska, Saso Gelev, Maja Kukuseva, *Faculty of Electrical engineering, University of Goce Delcev, Stip, Macedonia*

Sadržaj: *U ovom radu predlaže se uvuđenje modifikacije OSPF protokola, nekom vrstom komandi protokola između susednih rutera za razmenu nekih kontrolnih podatoka. Ovaj rad opisuje implementaciju novog modula u NCTUNS simulatoru za prethodno predlozeni protokol rutiranja. Mrežni NCTUNS simulator i emulator je visoko doverljiv, i imogućava simulaciju uređaja i protokola koji se koriste u žičnim i bezžičnim mrežama. NCTUNS mrezni simulator koristi otvorenu sistem arhitekturu da bi se moduli protokola lakše dodali simulatoru. Sledstveno, izvršiće se komparacija predložene modifikacije OSPF protokola sa aktuerlnim OSPF protokolom rutiranja I na bazi toga pokazati poboljšanje performansi.*
Klučne reči- *rutiranja, modifikovan OSPF protokol, racunarske mreze, NCTUNS mrezni simulator.*

Abstract: *The paper suggests introducing modified OSPF protocol, with a kind of command protocol between neighboring routers for exchange of some control data. This paper describes the implementation of a new module in NCTUns simulator, for a previously proposed routing protocol. The NCTUNS network simulator and emulator is a high-fidelity and extensible network simulator capable of simulating various devices and protocols used in both wired and wireless networks. It uses an open-system architecture to enable protocol modules to be easily added to the simulator.*
Consequently, the implementation of the proposed routing protocol is expected to enable comparison of the proposed routing protocol with current routing protocols and eventually show performance improvement.
Keywords - *routing, modified OSPF protocol, computer networks, NCTUNS network simulator.*

1. INTRODUCTION

In datagram networks, including IP networks, the process of routing is an issue for every packet. In process of routing router needs to be able to look at the packet's destination address and then to determine which of the output ports is the best choice to get the packet to that address. The forwarding process consists of receiving a packet, looking at its destination address, consulting a table, and sending the packet in a direction determined by that table. This is the traditional way of forwarding packets, and some improvements can be considered. One idea and realization proposed by L. G. Roberts [1] is called flow management.

The main proposal described in detail in [2] is the idea that the sending router could instruct its next hop neighbor to directly forward the packet on, without previously processing it, if the needed information is available in the current's router flow table [2]. The flow based exchange system among neighbor routers is based on several concepts:

- Flow identification (Flow ID) transmitted as part of IP header proposed for ipv4 and ipv6
- Flow Data Table (FDT) holding flow and router data for faster routing decisions
- Egress interface port to be used at neighbor router for specific flow
- New message type proposed for OSPF routing protocol used for exchanging FDT data.

2. MODIFICATIONS OF OSPF

The basic idea for the proposed modifications of the OSPF routing protocol lies behind the idea to utilize formerly made routing decisions considering specific flows of the current router, and the knowledge of routing decisions made by neighboring routers, as well. One of the modifications for OSPF involves using Flow IDs. The Flow ID of each packet will be kept inside the packet header. The Flow ID would be generated by the source host, and all packets for that flow that will be generated from the same source host will be carry the same Flow ID. The Flow ID identifies the 5-tuple (source IP address, destination IP address, source port, destination port, transport protocol). The packet flow data with unique Flow ID would be organized in a Flow Data Table (FDT). An example FDT is shown in Table 1.

Table 1 Example Flow Data Table (FDT)

Flow ID	SrcIP	DstIP	CurrRouIfc	NeighRouIfc
10345	192.168.0.10	192.168.1.5	If0	If15
10346	192.168.0.20	192.168.2.8	If3	If3

As shown in Table 1, the FDT keeps the following information:

- Flow ID is generated by the source host
- SrcIP- IP address of the source host
- DstIP-IP address of destination host
- CurrRIfc – egress interface of the current router that the last known packet from the specified flow was transmitted on.
- NeighRouIfc – egress interface of the neigbour router that the last known packet from the specified flow was transmitted on (next hop at the neighbor router for specific flow).

The OSPF protocol currently defines five types of messages: Hello, Database Description, Link State Request, Link State Update and Link State Acknowledgment. We propose a new message type with type value 6 called Flow Information Update Message (FIUM). The data field of the message would include the first four fields from FDT (Table 1) for a specific flow as shown in Table 2.

Table 2 OSPF Flow Information Update Message (FIUM) Data Field Format

16 bit	32 bit (IPv4)	32 bit (IPv4)	8 bit
Flow ID	**SrcIP**	**DstIP**	**CurrRIfc**

This type of message should be sent by the router every time the FDT is changed. The Flow Data Table can be changed in two situations: when certain packet is processed using the standard procedure, or when changes occur in the routing table. All routers receiving the message make corresponding changes to their FDTs regarding that specific Flow ID. The proposed modified OSPF routing protocol must consider the changes made the to the router's existent IP routing table, as well. This is needed in case there are some intermittent network topology changes recorded by the present OSPF routing protocol.

.

3. NCTUNS NETWORK SIMULATOR

NCTUns uses the real-life Linux's TCP/IP protocol stack to generate high-fidelity simulation results. NCTUns provides a highly-integrated and professional GUI environment in which a user can easily conduct network simulations. The NCTUns GUI program is capable of:

- Drawing network topologies
- Configuring the protocol modules used inside a node
- Configuring the parameter values used inside a protocol module.
- Plotting network performance graphs
- Playing back the animation of a logged packet transfer trace.

3.1 Components and architecture of NCTUns

NCTUns adopts a distributed architecture. It is a system comprising eight components [6], shown in Figure 1:

1. The first component is a GUI program by witch user edits network topology, configures protocol modules used inside the node, plots performance graphs, plays back the animation of a packet transfer trace etc.
2. Simulation engine program is the program which provides basic and useful simulation services to protocol modules. We call a machine on which a simulation engine program resides a "simulation server".
3. The third component is the set of various protocol modules, each of which implements a specific protocol or function (packet scheduling or buffer management). All protocol modules are C++ classes and are compiled and linked with the simulation engine program.
4. The fourth component is the simulation job dispatcher program that can simultaneously manage and use multiple simulation servers to increase the aggregate simulation output.
5. The fifth component is the coordinator program. On every simulation server, the "coordinator" program must be run up. The coordinator should be alive as long as the simulation server is alive.
6. The sixth component is the kernel patches that need to be implemented to the kernel source code so that a simulation engine process can run on a UNIX machine correctly.
7. The seventh component is the various user-level daemons that are run for the whole simulation.
8. The eighth part of NCTUns is composed of the real-life user-level application programs.

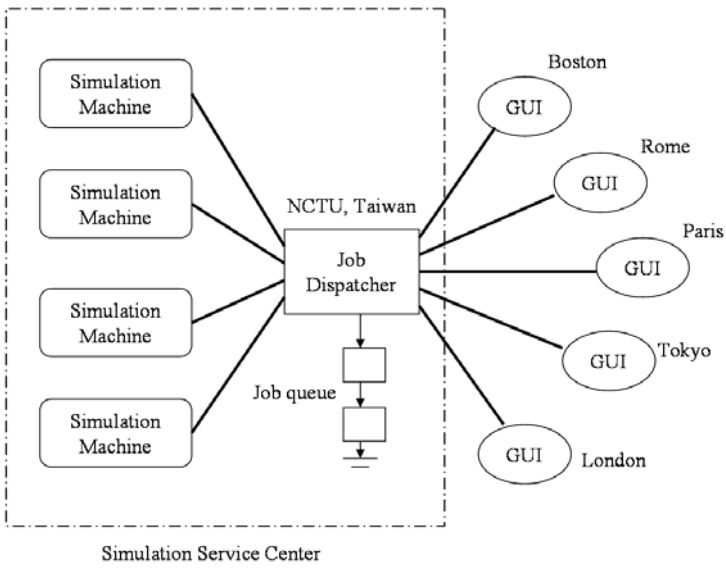

Fig. 1 The distributed architecture of NCTUns

3.2 Implementation of FIUM in NCTUns

The basic initiative for the proposed FIUM message of the basic OSPF routing protocol lies behind the idea to utilize

formerly made routing decisions considering specific flows of the current router, and the knowledge of routing decisions made by neighboring routers, as well. The main idea behind maintaining the information from FDT table is to enable making faster decisions at network level, without examining the existing IP routing tables. As previously mentioned, a new message type with type value 6 called Flow Information Update Message (FIUM), is proposed for using in the modified OSPF protocol. This message can be sent in several cases, as would be shown further on. The implementation of this new type of OSPF message in NCTUns would be using the existing OSPF implementation, with needed modifications.

The message number type (with value 6) will be defined in the header file for OSPF routing protocol, ospfd.h (placed in /nctuns/src/tools/misc).

Figure 2 explains the procedure for processing the FIUM message type. The FIUM message is exchanged between neighbor routers and is used for building the FDT table. When the message is received at one of the interfaces (whose IP address can be determined from the Neighbor structure containing all neighbors, defined in ospfd.h), the first step is to search the FDT table. The lookup in FDT is performed according to three parameters: *flow_id*, *src_ip* which are part of FIUM message, and *cuurent_ifc* determined as the interface on which the FIUM message has been received. If the FDT table search does not return any records according to these parameters, specifically the return argument *fdt_entry* is *NULL*, then the function add_entry() is being used, since the current router has no information for the flow described in the FIUM message and from that neighbor. The function add_entry() adds a new record in the FDT table copying the fields from the FIUM in this particular way: *flow_id*, *src_ip* and *dest_ip* from FIUM are copied as *flow_id*, *src_ip* and *dest_ip* in FDT; for *NeighRouIfc* in FDT we set *CurrRIfc* from FIUM, and for *CurrRouIfc* in FDT we set the interface on which the current router received the FIUM message. After calling add_entry() function, new FIUM message will be sent to all other neighbors except to the interface on which the previous FIUM message has been received. If the return result from the FDT search function is *fdt_entry != NULL*, meaning there **is such record** in the FDT table (same flow id, same source IP, and received from same neighbor), then all that needs to be done, is update that FDT entry with the new *CurrRIfc* from FIUM as the *NeighRouIfc* in FDT. To perform this step the function update_entry() is called. This situation can happen if there is some kind of change at the neighbor router, and that neighbor router changed **its** outgoing interface for packets from the designated flow. After this step of the procedure, again as in the previous case, another FIUM message is sent from the current router to all neighbors except to the interface that received the FIUM message.

Functions used as manipulation methods for searching, adding, updating and deleting entries in FDT table are defined in ospfd.h, and the entire procedure for receiving/sending FIUM messages is implemented in the NCTUns OSPF code – ospfd.cc.

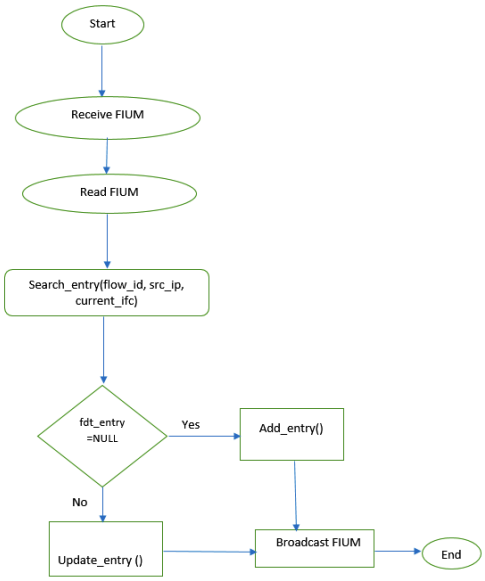

Fig. 2 Processing of FIUM message

4. RESULTS

In the end, we create network topology for testing of the modificated routing daemon. The network topology is formed of two routers shown on the picture.

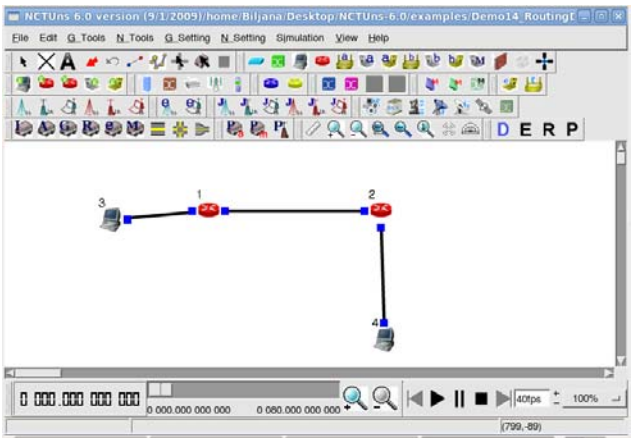

Fig. 3 Network topogy with two routers

Router marked on the picture with number 1 has two interfaces with IP addresses 1.0.2.1 (Port ID 2) and 1.0.1.1 (Port ID 1). Router marked with number 2 has this interfaces 1.0.1.2 (Port ID 1) and 1.0.3.1 (Port ID 2). Hosts marked with 3 and 4 have this ip addresses 1.0.2.2 (source host) and 1.0.3.2 (destination host). After creating network topology on each of the router we start an executive file ospfd. After completed simulation this results are shown.

In Table 3 are the basic information about FDT table set according to the first router as shown in the Fig. 3. These arguments are entered manually in the FDT table as defaults.

In Table 4 are the basic information about FDT table set according to the second router as shown on the image.These arguments are entered manually in the FDT table as defaults**.**

Specifying these parameters is enabled by the software simulator.

Table 3 FDT table on the router 1

Flow_id	Src_ip	Dest_ip	Current_ifc	Next_ifc
11223	1.0.2.2	1.0.3.2	1	2
11224	1.0.2.2	1.0.3.2	1	2

Table 4 FDT table on the router 2

Flow_id	Src_ip	Dest_ip	Current_ifc	Next_ifc
11225	1.0.3.2	1.0.2.2	1	2
11226	1.0.2.2	1.0.3.2	2	

After the neighboring routers exchange their FIUM messages, each of the routers will have new FDT tables, with data according to the used functions for adding, deleting and updating of the data. The first router will have the following FDT table.

Table 5 FDT table on the router 1

Flow_id	Src_ip	Dest_ip	Current_ifc	Dest_ifc
11223	1.0.2.2	1.0.3.2	1	2
11224	1.0.2.2	1.0.3.2	1	2
11225	1.0.3.2	1.0.2.2	2	
11226	1.0.2.2	1.0.3.2	1	2

In the next exchange FDT tables between neighboring routers, routers 1 and 2 will update their FDT tables and will be called stable, that will have information about the output interface of a neighboring router. The router 2 will have the following FDT table.

Table 6 FDT table on the router 2

Flow_id	Src_ip	Dest_ip	Current_ifc	Dest_ifc
11225	1.0.3.2	1.0.2.2	1	2
11226	1.0.2.2	1.0.3.2	2	
11223	1.0.2.2	1.0.3.2	2	
11224	1.0.2.2	1.0.3.2	2	

5. CONCLUSION

As mentioned at the beginning of this paper, the primary motivation behind the modifications of OSPF was to enable increase in speed over the current OSPF routing protocol. With the proposed modifications we expect greater speeds in packet forwarding, enabling the router receiving the packet to forward directly to the designated interface without any processing.

The implementation of this modified OSPF routing protocol, as a protocol module in NCTUns is important for research, testing and development of this proposed routing protocol. With the successful implementation of this modified OSPF protocol, we can show that NCTUns network simulator can be easily upgraded with new modules, and then used to compare the existing routing protocols with the modified OSPF protocol.

REFERENCES

[1] Roberts, L.G. ”A radical new router”, Spectrum IEEE, July 2009, pp. 34-39.

[2] Marija Kalendar, Aristotel Tentov, Danijela Jakimovska, Goce Dokoski, “A Novel Flow Based Approach In Routing Decision Mechanism”, Proceedings of 3rd IASTED International Multi-conference on Automation, Control, and Information Technology (ACIT2010), 15-18 June 2010, Novosibirsk, Russia, pages: 158-163

[3] L. Peterson and B. Davie, “Computer Networks: A Systems Approach”, Morgan Kaufmann Publishers, San Francisco, CA, 4th Edition, 2007.

[4] D. Medhi, K. Ramasamy, ”Network Routing: algorithms, protocols, and architectures”, Morgan Kaufmann Publishers, 2007.

[5] S.Y. Wang and H.T Kung, “A Simple Methodology for Constructing Extensible and High-Fidelity TCP/IP Network Simulators”, INFOCOM‟99, March 21-25, 1999, New York, USA.

[6] S.Y. Wang, C.L. Chou, C.C. Lin, “The Design and Implementation of the NCTUns Network Simulation Engine”, Simulation Modelling Practice and Theory, 15 (2007) 57-81.

[7] Shie-Yuan Wang, Chih-Liang Chou, and Chih-Che Lin, “The GUI User Manual for the NCTUns 4.0 Network Simulator and Emulator”, July 2007 Produced and maintained by Network and System Laboratory, Department of Computer Science, National Chiao Tung University, Taiwan

[8] Shie-Yuan Wang, Chih-Hua Huang, Chih-Che Lin, Chih-Liang Chou, and Kuo-Chiang Liao, “The Protocol Developer Manual for the NCTUns 4.0 Network Simulator and Emulator”, July 2007, Produced and maintained by Network and System Laboratory, Department of Computer Science, National Chiao Tung University, Taiwan