MATEC Web of Conferences **44**, 01027 (2016)
DOI: 10.1051/matecconf/20164401027

The Study and Simulation of Multi-interface Extension for OLSR Protocol

Hui Zhong[1,a], Xu Bin Wang[2]

[1] *Shenyang Jianzhu University, Faculty of Information & Control Engineering, 110168, Shenyang, China*
[2] *Shenyang Jianzhu University, Faculty of Information & Control Engineering, 110168, Shenyang, China*

Abstract. The way to improve the performance of the throughput in the wireless Mesh network is nodes with multiple interfaces. In view of the problem of multi-interface multi-channel extension for OLSR protocol, multi-interface multi-channel OLSR protocol is proposed. Nodes can have multiple network interfaces, but each node has only one IP, using the index of interface to distinguish the multiple interfaces of the node. Selected the receiving interface according to the channel, modified the message delivery method and introduces the protocol implementation and technology. Finally, the simulation is carried out by using the extended NS2. Simulation results show that extended OLSR protocol has high network performance, and improve the network throughput. What is new and original in this paper is adding multiple physical interface support for proactive protocol OLSR.

1 Introductions

Wireless Mesh Network (WMN) [1], has the characteristics of high performance ratio, scalability, and high reliability, etc. However, in general, the nodes in the network have only one single interface. Single interface affects the data transmission between nodes, which limits the throughput of the network. One of the most effective ways to solve the problem is to equip nodes with multiple interfaces [2]. Wireless Mesh networks with multiple interfaces can improve the data processing speed of nodes; improve the network throughput .

In order to achieve the goal, it is necessary to expand the NS2. The well-known scheme is the method of Ramon multi-interface extension in the AODV protocol [3]. However, there is no extension of the OLSR protocol. In this paper, we refer to the method of the Ramon scheme and modify it, then expand the OLSR protocol to the multi-interface multi-channel OLSR.

2 NS2 multi-interface multi-channel expansion

It needs to modify the MobileNode model in order to extend the NS2 to multi-interface multi-channel [4]. The original MobileNode model has only one component, which is the link layer, ARP, interface queue, MAC, network interface and channel. The modified MobileNode model for each interface has a set of components, as shown in figure 1. The number of interfaces and the distribution of the channels can be easily controlled in the simulation scripts.

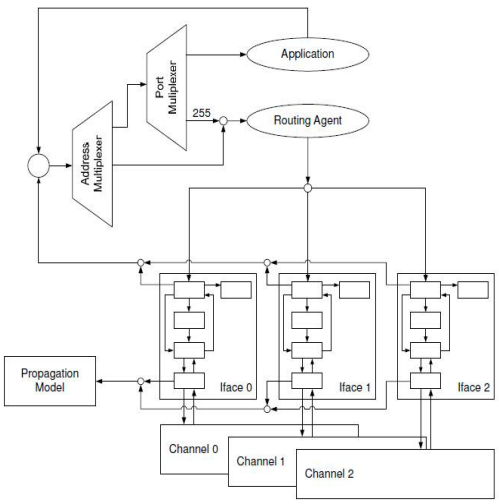

Figure 1. Modified MobileNode architecture.

3 Multi-interface multi-channel AODV

Ramon scheme is extended the AODV protocol under NS2. The main idea of the scheme is to increase the number of interfaces and channels, increase the number of channels under the AODV protocol, and modify the protocol itself, so that AODV can support multi-interface [5].

Ramon scheme has been modified in three ways, which is to modify Tcl, modify C++ code and modify AODV routing agent. The modification of Tcl is mainly to complete the expansion of MobileNode model, and the modification of C++ file is used to adapt to the new framework, and most of the changes are affecting the simulator to deal with the MobileNode class. The last part

[a] Corresponding author: syzhonghui@sina.com

Article available at http://www.matec-conferences.org or http://dx.doi.org/10.1051/matecconf/20164401027

of the change is the implementation of the AODV protocol. The way to transmit message (RREP, RRER, RREQ and HELLO) is different between Multi-interface AODV and the original AODV, and it also needs to add some new parameters to the routing agent to manage the corresponding interface information. So this part has made some changes from two aspects: routing finding and routing maintenance [6].

With the increase of the number of effective network interfaces, the multi-interface multi-channel AODV, which improves the throughput of the network, reduces the end-to-end delay, and the network also becomes more stable [7].

4 OLSR protocol

OLSR routing protocol is a kind of the optimal link state routing protocol which is driven by the MANET IETF (Mobile Ad hoc Net-work) working group, especially suitable for large and scale mobile network [8].

The operation of the OLSR protocol relies on the various types of data table of the nodes[9]. The node broadcasts HELLO messages periodically, and forms a Link Set based on the HELLO messages, then forms a Neighbor Set based on the Link Set. The TC messages are broadcasted to the whole network by MPR nodes. Each node maintains a Topology Set. If the node contains multiple interfaces (each interface is only assigned one IP address), an interface Association Set table is formed. Finally, the routing table of the node is obtained by calculating these data tables.

Interface Association Set

I_iface_addr	I_main_addr	I_time

Figure 2. Interface Association Set.

This set is used to store the relationship between the main address of the node and the OLSR interface addresses of a node. I_iface_addr represents t the OLSR interface addresses, I_main_addr is the main address of the node, and I_time represents the effective time of the event.

Link Set

L_local_iface_addr	L_neighbor_iface_addr	L_SYM_time	L_ASYM_time	L_time

Figure 3. Link Set.

Link Set is used to store the link information of the node. L_local_iface_addr represents the local interface address of the link, L_neighbor_iface_addr represents the link's neighbor interface address, and L_SYM_time represents the time when the link is symmetry, L_ASYM_time indicating that the time when the link is asymmetric, and L_time is an effective time for the event.

Neighbor Set

N_neighbor_main_addr	N_status	N_willingness

Figure 4. Neighbor Set.

This set is used to store information about a neighbor node. N_neighbor_main_addr for the main address of the neighbor node. N_status is the state of a neighbor node. N_willingness for the forwarding value of the neighbor node.

2-hop Neighbor Set

N_neighbor_main_addr	N_2hop_addr	N_time

Figure 5. 2-hop Neighbor Set.

This set stores the information of the 2-hop neighbor nodes. N_neighbor_main_addr represents the main address of the neighbor node. N_2hop_addr represents the main address of the 2-hop neighbor node. N_time is a valid time for the event.

Topology Set

T_dest_addr	T_last_addr	T_seq	T_time

Figure 6. Topology Set.

By exchanging TC messages periodicity, the topology set recorded the network topology. T_dest_addr represents the main address of the destination node, T_last_addr is the main address of the last hop of the destination node, T_seq is the sequence number of the event, and T_time is the effective time of the event.

Routing Set

R_dest_addr	R_next_addr	R_dist	R_iface_addr

Figure 7. Routing Set.

R_dest_addr is the interface address of the destination node, R_next_addr is the interface address of the next hop neighbor node, R_dist is the number of hops to the destination node, R_iface_addr is the address of the local interface to the next hop.

5 Multi-interface multi-channel OLSR extension scheme

OLSR is a proactive protocol, each node in the OLSR network needs to maintain a routing table. Therefore, the routing table is the extraction of the link state, neighbor state and topology information. That is to say, when the corresponding event is changed, the routing table also needs to be calculated.

Node builds and updates their own network topology by exchanging control messages periodically. Among them, there are three types of control messages. First, the HELLO message, which is used for link sensing and populating neighbor; Second, the TC message, which is used to declare the topology; Third, the MID message is used to declare the relationship between OLSR interface addresses and main addresses. The Interface Association Set is formed according to the MID message, and the Topology Set is formed according to the TC message. The formation of the Link Set depends on the HELLO message. After the formation of the Link Set, the neighbor populating process is completed according to the Link Set and the HELLO messages, format the Neighbor Set and 2-hop Neighbor Set.

In order to realize the multi-interface multi-channel extension of OLSR protocol, a new approach is proposed, which a node can have multiple interfaces. But each node

has only one IP address. Which interface is used to send and receive messages is distinguished by the index of the interface. Since each node has only one IP address, MID message will be lost.

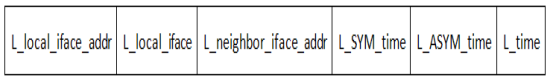

Figure 8. Modified Link Set.

The main process of link sensing is to maintain a local Link Set using HELLO messages that are exchanged between nodes periodically. When a node has multiple interfaces (only one IP address), it needs to know which interface the node has is formed a link with its neighbor. Therefore, the paper is expanded based on the original Link Set, add an index of local interface L_local_iface after the L_local_iface_addr, and record the interface which is used, as shown in Figure 8.

Routing tables are also involved in the Neighbor Set, 2-hop Neighbor Set and Topology Set. The Neighbor Set and 2-hop Neighbor Set is forms by the Link Set and HELLO message. Each node maintains a Link Set, we need to add the index of interface to the Routing Set when the routing table is calculating, the Neighbor Set and the 2-hop Link Set do not need to change. Topology Set shows that the topology of the entire network, and do not need to save the interface information, so it is not necessary to modify it.

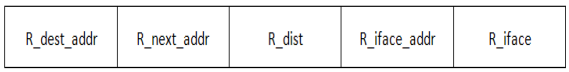

Figure 9. Modified Routing Set.

In order to enable nodes to transmit data with the the added interface, an interface index is required to added during the calculation of the routing table. The protocol only needs to know which interface the data need to be sent from, so we add the index of local interface R_iface into the routing table after the local interface address, as shown in Figure 9. Then we can get the corresponding index by looking up the Link Set.

6 Multi-interface multi-channel OLSR implementation

Based on the above-mentioned programme, the NS2 related components can be modified. In order to make NS2 more suitable for multi-interface multi-channel model, we have modified the Tcl script, C++ source code and routing protocol.

6.1 Modify the Tcl script

The first needs to make changes to the tcl/lib/ns-lib.tcl file and tcl/lib/ns-mobilenode.tcl file. In ns-lib.tcl, the code in the original NS2 is not defined the corresponding multi-interface multi-channel parameters, which only supports single interface single channel, thus creating the four new program to achieve the goal. The change-numifs creates an interface for the node; add-channel adds the corresponding channel for the interface. The ifNum process is used to set the number of interfaces, and get-numifs is used to get the number of interfaces. In addition, it is also required to initialize the corresponding variables in the node configuration process.

For ns-mobilenode.tcl, the first need to modify the add-target, call get-numifs to get the number of interface, while the process of routing agent if-queue has been modified to adapt to the multi interface. Secondly, it needs to modify the add-target-rtagent. This program uses the get-numifs to get the number of interfaces, and then uses the variable to connect the routing agent and the link layer. The next one need to be modified is add-interface. Since the previous process created one ARP per node, we need to create one ARP per interface, so the arptable_ is modified as an array arptable_ ($i). The last change affects the way the MobileNode is created and reset.

6.2 Modify the C++ source code

The purpose of this section is to adapt the new framework. Most of the changes affect how the simulator deals with the MobileNode class.

After creating the multi-interface architecture for a mobile node, it is required to connect to the appropriate channel properly. Therefore, the modification in Mobilenode.h is to manage the channel with two pointers nextX_[MAX_CHAN] and prevX_[MAX_CHAN]. However, after the redefinition of prevX_ and nextX_ the getLoc function errors, so we redefinition getLoc method. In addition, we need to add a new definition of getLoc in mobilenode.cc file.

Since the previous definition of two array pointers need to be used in the channel.cc file in order to manage the list of nodes, it is necessary to modify the nextX_ to nextX_[this->index()] and the prevX_ to prevX_[this->index()] through the entire file. In addition, when a packet is sent, the first need to evaluate is which node is close enough to the source node, and is connected to the channel. Then packets can be sent to the interface of node. This is meaningless, however, because the packets can only be received from the interface with the appropriate channel. In this case, we need to inspect whether the sender and the receiver of the interface is using the same channel. "if (rifp->channel () ==this) " was added to the code.

The last one need to be modified is the mac-802_11.cc. In order to recognize the interface, we need to add " hdr->iface ()=addr ();"into it.

6.3 Modify the routing protocol

The first two parts of the changes is to do the preparation for using the multi-interface, routing protocol itself is also modified to adapt to this change. In the OLSR_repositories.cc, the iface_ and the Iface () are added to the struct OLSR_rt_entry. The index of local interface is saved in the variable iface_ in the routing table, and the return value of the function iface() is iface_. In the struct OLSR_link_tuple, added the variable local_iface_ and the function local_iface() , which is used to store the interface index of the local node.

In the OLSR_state.cc, the operation of Link Set is required to add a function find_link_tuple, so the Link Set entries corresponding to the interface address and

interface index are found by using this function, and then return it.

In the OLSR_rtable.cc, we need to modify the add_entry method. Add interface index entry->iface() = iface_ after the local interface address. Since the original add_entry method using the map container, the key and value is one to one relationship, so we change the map to multimap. This allows for a destination address with multiple routing tables that can be present.

In order to achieve the OLSR protocol interface extensions, we need to change the routing proxy implementation. In OLSR.h , the first one need to declare is a global variable numIfaces. It used to store the number of interfaces owned by the node. Now we need to declare two arrays: targetlist[] and ifqueuelist[]. The targetlist[] is used to store all the interfaces of the link layer model in a particular node, the ifqueuelist[] is used to store the corresponding queue. At the same time, it also needs to define a new constant MAX_IFACE, which is used to manage those two arrays.

In the OLSR.cc, it needs to modify the command method of the OLSR routing agent class, and completes the initialization of the above variables. In order to use the Tcl script in the simulation, we also need to implement the distribution of Tcl commands.

The role of the method send_pkt is to generate enough OLSR packets according to the cached OLSR message, and then send it out. Since the node may contain two or more interfaces, it is required to determine whether the node has multi-interface. If it is multi-interface, the OLSR packets are broadcast on all interfaces. Otherwise, keep the original behavior unchanged.

Method recv_olsr is used to process the received OLSR packets. In the recv_olsr method, we need to define a local variable recv_iface. This variable is used to store the index of the interface that used by the node to receive packets. According to the index, we can learn which interface is connected with its neighbor node. Therefore, in the case of multi-interface, we need to calculate the index. We can use the recv_iface=ch->iface()-((Mac *)ifqueuelist[0]->target())->addr() to calculate it. Among them, the ch->iface ()is the number of interface which receive packets, and the second part is the first number of interface. In the case of a single interface, the index is not required, so we set recv_iface to -1, an invalid value. After we get the index of the receiving interface, we pass the index to the process_hello method.

Method process_hello is used for processing HELLO message, including link sensing, populate neighbor and 2-hop neighbor, and MPR computation. The index recv_iface will be used for the link sensing process, and finally form the Link Set. In the Link Set, add the local interface index local_iface () after the local interface address, and assign recv_iface to it.

When the destination address of the packets received by the node is not itself, the node calls the forward_data method to forward the packet. Different with send_pkt method, the destination address of the forwarding packets have already been learned, so the forwarding is unicast. Lookup the routing table, get the interface address to the next hop and the index, and then put packets into the scheduler.

The calculation of the routing table contains in rtable_computation method. Under the multi-interface extension, modification Link Set of will take effect. We get the index of the interface in Link Set, add it to routing table by add_entry method, and select the interface when the packet is sent.

7 The simulation and results of experiment analysis

In order to verify the multi-interface multi-channel expansion, we need to write Tcl script for simulation. Then the simulations compare single-interface OLSR with multi-interface OLSR, and multi-interface OLSR with multi-interface AODV.

Simulation 1: single-interface OLSR and multi-interface OLSR.

Create such a scene, 5 wireless nodes, numbered from 0 to 4. Node 0 sends packets to node 4 by nodes 1, 2 and 3. The single-interface OLSR protocol created only one channel per node, while the multi-interface OLSR protocol creates two different channels per node. Using the UDP connection and CBR flows, and packet size is 500 bytes, send interval is 0.005 seconds, the simulation lasts 60 seconds.

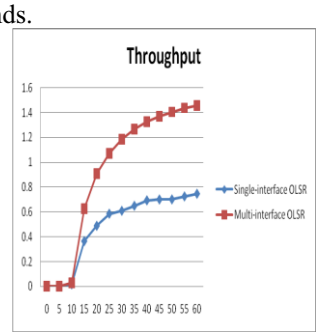

Figure 10. Throughput.

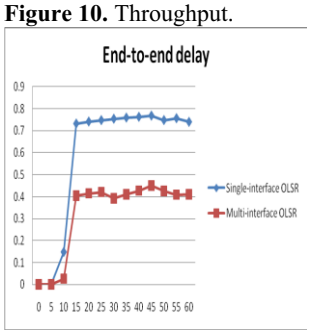

Figure 11. End-to-end delay.

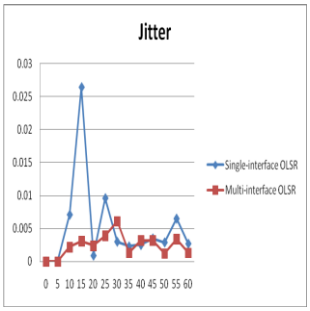

Figure 12. Jitter

Throughput is the number of successful delivery of data within a unit time. Figure 10 shows that with the start of simulation, the throughput of the single-interface and multi-interface OLSR protocols are all start to rise, and the rise of the extended multi-interface OLSR protocol is significantly faster. This indicates that the throughput of the extended OLSR protocol is better than single-interface OLSR.

End-to-end delay is the delay time of the CBR packet to the end. From Figure 11 it can be seen that after 10 seconds the end to end delay is basically stable, and the

extended OLSR protocol is lower than the original OLSR protocol.

Jitter means the delay variance. Because of the state of network changes and the traffic is always changing. So the data may be queued in the queue, waiting for transmission. The time of each packet from the source node to the destination node is not necessarily the same. This time difference is Jitter. As can be seen from Figure 12, the jitter of the single interface OLSR protocol is much bigger than the multi-interface OLSR, so it is shown that the multi-interface OLSR protocol is more stable than the original OLSR protocol.

Simulation 2: multi-interface OLSR and multi-interface AODV.

To build a simulation with 5 nodes, node 0 send data to node 4 by nodes 1, 2 and 3. Both two scripts are used the multi-interface protocol, and each node creates two different channels. Use the UDP connection and the CBR flows, the packet size is 500 bytes, send interval is 0.005 seconds; the simulation time is 60 seconds. Nodes start sending and receiving data from 1.0 seconds.

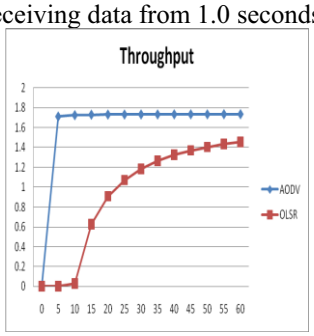

Figure 13. Throughput.

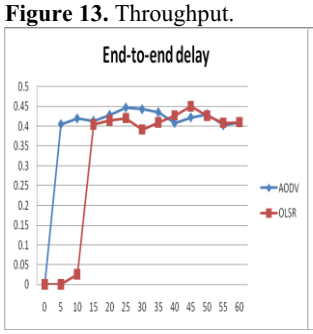

Figure 14. End-to-end delay.

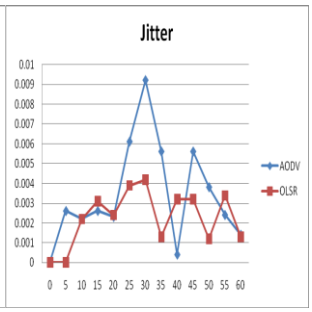

Figure 15. Jitter.

As s By Figure 13, in the same scenario, Throughput of multi-interface AODV is growing rapidly, and the multi-interface OLSR protocol is slowly improving. This may have relations with the network environment, the number of nodes and the different operation modes of the reactive routing protocols and the prior routing protocols.. In the case of small number of nodes, the throughput of multi-interface AODV is better than the multi-interface OLSR.

As can be seen from Figure 14, with the beginning of the simulation, the end to end delay of the two protocols are beginning to rise, and eventually stabilized in a range. It can be known that, in the case of small number of nodes, the performance of multi interface OLSR and multi interface AODV is similar.

As shown in Figure 15, in most of the time, the jitter of the multi-interface AODV is bigger than the multi-interface OLSR. Therefore, it can be concluded that the stability of the multi-interface OLSR protocol is better than multi-interface AODV.

8 Conclusions

In recent years, the research of multi-interface multi-channel wireless Mesh networks has been an active research topic. In NS2, the simulation of wireless network is a very complex process, it is necessary to understand the working mechanism of each component in the simulator, which requires a lot of time and work. The original OLSR defined the multiple network interfaces, but because of the characteristics of the simulator architecture, it cannot be used. This paper presents a method to implement multi-interfaces and multi-channels on NS2, complete the multi-interface multi-channel OLSR protocol, and improve the throughput of wireless network. However, the existing interface switching strategy is not very good; it can't fully play the advantages of multi-interface and multi-channel. Therefore, the next step will be to make further improvements to the OLSR, so that it can play the advantages of multi-interface performance better.

References

1. XU Tong, YANG Shou-bao, HU Yun. P2P Super Node Selection Strategy Based on Physical Characteristic of WMN[J]. Computer Engineering, 2009, 35(1):101-104.
2. ZHONG Hui;LANG Bing. Research on Link Redundancy Abstract Model for Multi-radio Mesh Network [J]. Journal of Chinese Computer Systems, 36(6):1255-1260.
3. Calvo R A, Campo J P. Adding multiple interface support in NS-2[J]. University of Cantabria, 2007.
4. XIE Peng-yu,SONG Ling,CHEN Yan. Research of AODV Extension for Multi-interface Multi-channel WMN[J]. Computer Engineering, 2011, 37(13): 74-76.
5. YOU Xiao-qian, WEI Yong-lin. (2009).AODV Routing Protocol Supporting Multi-interface. Computer Engineering,35(23), 109-111.
6. FU Qi, CHEN Zhi-gang,JIANG Yun-xia. (2013). Research on Multi-interface Multi-channel Simulation Model Extension Based on NS-2. Computer Engineering, 39(4), 113-117.
7. LAI Cai-hua, TAO Yang. Research on multipath routing algorithm in Wireless Mesh Networks [J]. GUANGDONG Communication Technology, 2010, 30:10-14.
8. ZHAO Jian, SUN Jun-suo. (2008). Simulation and Analysis of an improved OLSR Routing Protocol Based on NS2 . Computer Simulation, 25, 161-163.
9. CHEN Ji-tong. (2006). Design and implementation scheme of OLSR routing protocol for multi-hop wireless ad hoc networks [D](Doctoral dissertation, University of Electronic Science and Technology of China).