# Introducing a Distance Vector Routing Protocol for ns-3 Simulator

Janaka Wijekoon
Hiroaki Nishi Laboratory, Keio University, Japan
janaka@west.sd.keio.ac.jp

Rajitha Tennekoon
Hiroaki Nishi Laboratory, Keio University, Japan
rajitha@west.sd.keio.ac.jp

Erwin Harahap
Hiroaki Nishi Laboratory, Keio University, Japan
erwin2h@west.sd.keio.ac.jp

Hiroaki Nishi
Dept. of Systems Design Engineering Keio Univ.
west@sd.keio.ac.jp

## ABSTRACT

In network research, network simulators have been shown to be useful for testing and changing network protocols by means of a controlled manner. As an Internet systems simulator, ns-3 simulator provides an ideal simulation environment for network research. However, ns-3 is lack of table-driven IPv4 routing modules. Despite the fact that a routing protocol is a mandatory component of a network, it is necessary to introduce a table-driven routing protocol for the ns-3 which is able to maintain the network connectivity and consistency. To this end, we introduce a distance vector IPv4 wired routing protocol for the ns-3 simulator (DVRP). The proposing protocol is developed as a table-driven wired routing protocol. In this study, we describe the proposing routing protocol, including its design, implementation, behavior on networks, and limitations.

## Categories and Subject Descriptors

C.2.6 [**Internetworking**]: Routers; C.2.2 [**Network Protocols**]: Routing Protocols

## General Terms

Algorithms, Design

## Keywords

ns-3, routing protocol, distance vector routing, network simulation, wired routing protocols.

## 1. INTRODUCTION

Network researchers must test and evaluate numerous network protocols in controlled environments to determine whether such protocols are robust and reliable. Because the Internet is the largest production-scale network, it is impractical to conduct controlled experiments directly on the Internet. Therefore, simulation software is a vital tool in developing, testing, debugging, and evaluating network protocols. Nevertheless, simulation software should be able to simulate a particular network's behavior and, should be able to provide analysis tools to evaluate results. In the study of [3], Carniro et al. stated that the ns-2 [8] is the most used simulator among network researchers. As the successor of the ns-2 simulator [8], the ns-3 simulator [21] was introduced for more realistic simulations of network protocols.

The ns-3 simulator [21] is a discrete-event simulator, and was proposed as the eventual replacement of the ns-2 simulator [8]. The ns-3 simulator uses *ns3::NetDevices* to connect *ns-3::Nodes*, because the *ns3::NetDevices* emulate physical network cards and the *ns-3::NetDevices* possess both Layer-3 IP and Layer-2 MAC addresses. Two *ns3::NetDevices* are connected using an *ns3::Channel*, which represents a network communication medium (wired or wireless links) [20]. Furthermore, ns-3 supports most generic real-world packet structures, and the packets are serialized and de-serialized as they traverse the network stack [3]. In addition, ns-3 supports both IPv4 and IPv6 addressing schemes. These features make ns-3 simulator an ideal simulator, which provides a more realistic and real-world-like simulation environment [19,20].

Despite its advantages, ns-3 is still a new simulator and new models are currently being developed and incorporated into ns-3 distributions [5,16–18,23,25,26,28]. Because ns-3 was introduced to simulate real-network-like simulations, ns-3 must support routing protocols for realistic evaluations of networks. A routing protocol is necessary to maintain routes despite changes in network connections [7,13]. In [2], Brakmo et al. theorized that most practical networks exhibit extremely complex behavior because of: 1) suitable protocol interaction, 2) complicated network topologies, and 3) complex traffic patterns. Moreover, according to [2,6,7,9,13,26], a routing protocol is directly related to the performance of a network. However, thus far, ns-3 does not comply with any Ipv4 wired routing protocol modules. This limitation limits the ability to simulate and analyze network topologies such as CDNs [22,27] on the ns-3 simulator.

To this end, in this study, we present the design and implementation of a table-driven IPv4 wired routing protocol for ns-3. The proposing protocol is developed as a distance vector routing protocol (DVRP) based on RFC 1058 [14] and RFC 2080 [11]. The DVRP uses some vital functions of the aforementioned RFCs to route and neighbor management. The ns-3 DVRP module supports: 1) neighbor discovery and management, 2) shortest path calculation based on the number of hops to a particular destination, 3) route management including processing periodic and triggered up-

.

SIMUTOOLS 2015, August 24-26, Athens, Greece
Copyright © 2015 ICST
DOI 10.4108/eai.24-8-2015.2260345

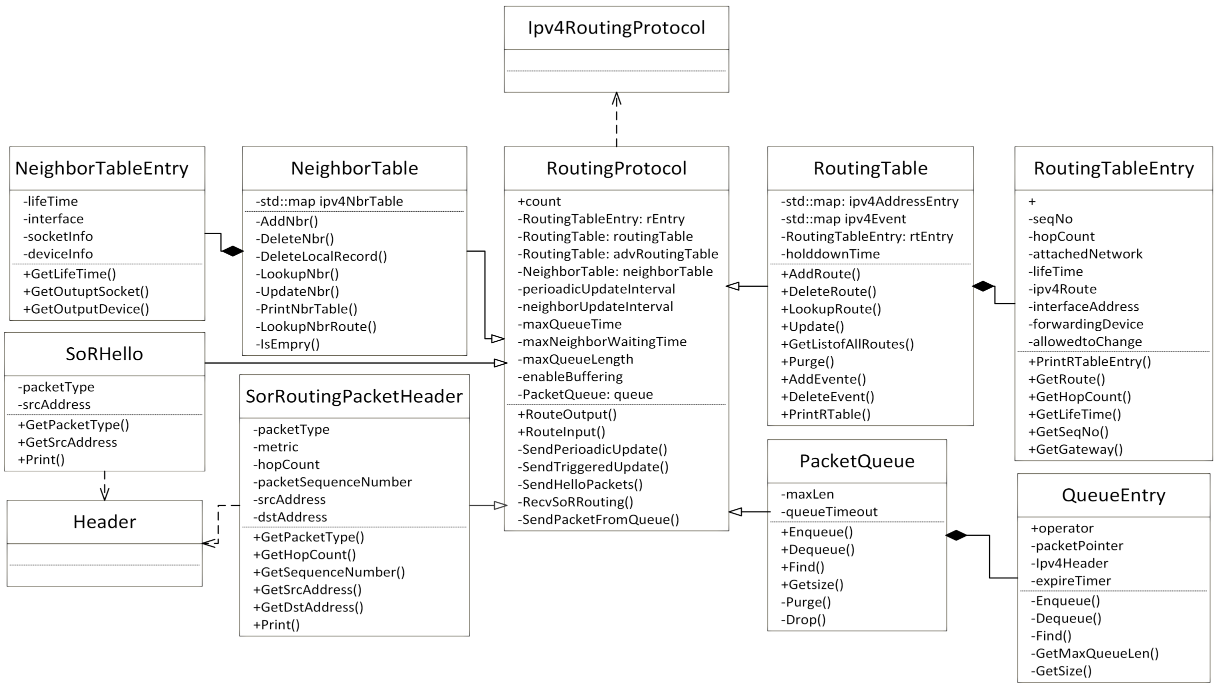

**Figure 1: DVRP Class Diagram**

date messages, and 4) per-node packet buffering to support consistent packet delivery. We compared the DVRP module with existing ns-3 IPv4 routing protocols and herein discuss the results.

The remainder of the paper is structured as follows. Section 2 explains the existing routing protocols implemented in ns-3. Section 3 describes the design of DVRP module. Section 4 provides the evaluation details of the proposed method. Section 5 concludes the study and suggests future studies.

## 2. EXISTING NS-3 ROUTING PROTOCOLS

The infrastructure of ns-3 and its design are intended to support both Ipv4 and Ipv6 routing protocols. ns-3 supports open source routing implementations and also facilitates research into novel routing techniques [5, 20, 21, 25]. Furthermore, ns-3 provides Direct Code Execution (DCE), which is a framework for ns-3 that provides environment to execute existing implementations (e.g., network protocols, applications and user-space protocols) without source code changes.

Literature reveals that the ns-3 simulator is popular among researchers who are experimenting on wireless mobile ad hoc networks (MANET) [15]. Consequently, with in recent years, several wireless routing protocols such as DSR [5], DSDV [18], AODV [18], and OLSR [1], have been introduced and tested on ns-3. Furthermore, the ns-3 is used by numerous researches to compare wireless routing protocols (e.g., compare OLSR and DSDV Protocols in a VANET Scenarios) [24].

When considering LAN/WAN implementations on the ns-3, in [17], Ashok Kumar et al. proposed ns-3 to simulate various architectures of data center networks, including both wired and wireless topologies. In [16], Junseok Kim argued that AODV is a proactive table-driven routing protocol, and its link usage is comparatively high. As a solution, they proposed a simple reactive on-demand routing protocol for Wi-Fi access points. However, thus far ns-3 is lack of wired routing protocol implementations. ns-3 only consist of reacting on-demand routing protocols [5, 6] (See Section 2.1).

### 2.1 Existing ns-3 wired routing protocols

The ns-3 simulator [21] essentially introduces three IPv4 connectivity methods for wired network simulations: 1) default/static Ipv4 routing protocol [20], 2) IPv4 global routing protocol [20] and, 3) IPv4 NIX-vector routing protocol [19]. These three routing protocols are reactive on-demand routing protocols [5, 6] IPv4 global and IPv4 NIX-vector routing protocols maintain per-node routing tables. However, those routing tables are built by analyzing the total topology only once at the beginning of a simulation [19, 20] and thus, both these protocols do not use a route advertisements to maintain up-to-date routing tables based on changes to a network topology. Besides, by using these packet forwarding methods, the ns-3 simulator can send packets between two nodes. Yet, simulate and analyze real networks such as a CDN [21] on the ns-3 simulator is impossible and thus represents a limitation for this simulator.

#### 2.1.1 Default / Static Routing Module

The ns-3 simulator uses the *ns3::IPv4StaticRouting* module as the basic connectivity method of small networks [20]. However, static routing is not a practical solution for route management on medium and large-scale networks [7].

#### 2.1.2 Global Routing Module

The *ns3::GlobalRoutingModule* is a reactive on-demand routing protocol [20]. At the beginning of a simulation, the module collects all static routes and stores them in routing tables. Upon packet arrival, routers that are configured for global routing then return the first route available in the routing table that matches the packet's destination. The returned route is not necessarily the shortest path for a particular destination. Therefore, packets might reach their destinations through a long distance path. In addition, a global routing module does not employ route update messages to maintain an up-to-date routing table according to the network changes.

### 2.1.3 NIX-Vector Routing Module

The *ns3::Ipv4NixVectorRouting* performs on-demand route computation using a breadth-first search and efficient route-storage data structure known as a NIX-vector [19]. When a packet is generated at the packet generation node, the *Ipv4NixVectorRouting* module first calculates the destination route. The created route is then stored in an NIX-vector which is attached to the generated packet. Routers use the NIX-vector to determine the next forwarding hop toward the packet's destination [19]. Similar to the global routing module, this module does not maintain routing updates to adapt network changes.

### 2.1.4 BRITE Routing Module

In addition to the aforementioned basic ns-3 connectivity methods, Swenson and Riley proposed a method to generate large topologies on an ns-3 simulator using a novel method called a BRITE routing topology generation tool [26]. They stated that to "maintain a proper routing protocol is mandatory for large network topologies." Therefore, they proposed an enhancement to the ns-3 global routing module to generate a per-node routing table and to calculate routes using a CUDA kernel. They showed that the CUDA-based routing module is faster than the ns-3 NIX vector routing protocol. Unfortunately, the proposed routing method is also not capable of route advertisement to adapt the topological changes.

### 2.1.5 RIPng Routing Module for IPv6

The RIPng module is the most recently introduced routing protocol to the ns-3. It was developed by T. Pecorella based on RFC 2080 [11] to the ns-3.20 [23]. This is the only fully functional routing module that ns-3 presents thus far. Unfortunately, as the RIPng module only supports IPv6 routing, it is not capable of simulating IPv4 routing. Therefore, in this study, this module was not compared to the DVRP module.

## 3. DVRP MODULE FOR NS-3

This section describes the design and implementation of the ns-3 DVRP module and focuses on: 1) neighbor discovery and maintenance, 2) route-table creation and maintenance, 3) route- and neighbor-table update mechanism, and 4) packet-buffering mechanism. All major attributes used in this implementation are listed in Table 1 and relationships among all classes are illustrated in Figure 1.

The ns-3 DVRP module is developed using the DVR algorithm [10]. The DVRP module uses the minimum hop path as the shortest path for a certain destination and the shortest path is calculated using the Bellman-Ford algorithm [4]. Routers calculate the shortest paths on update message arrival and the calculated shortest path is used to update the routing tables. Note that the current DVRP version is developed as a host-address-based routing protocol and thus maintains a host-address-based routing table instead of a network-address-based routing table.

The basic functions of the DVRP module can be explained as follows. For simplicity of explanation, we assume the following scenario: let $X$ be a router that receives a new destination $D$ from its neighbor $N$ at a distance of $c_N$. This means that$X$ can reach to $D$ through $N$ with the cost $c = c_D + c_N$. Consequently, the following rules were implemented to the DVRP for route-table maintenance and route-table advertisement processes. Note: the $s$ denotes the sequence number of the existing record and $s_{new}$ denotes the sequence number of the newly received record.

- if $D$ is a new destination, $X$ adds $D$ in to the routing table as $(D, viaN, c_D + c_N, s_{new})$
- if $D$ is a destination that is already in the routing table with $(D, M, c, s)$, where $c > c_D + c_N$ and $s > s_{new}$, then the routing entry will update for $(D, viaN, c_D + c_N, s_{new})$
- if $D$ is a destination that is already in the routing table with $(D, M, c, s)$, where $c > c_D + c_N$ and $s <= s_{new}$, then the routing entry will not be updated.
- the $X$ sends periodic updates to its neighbors about $D$ by setting $c = c + 1$ and $s_{new} = s + 2$
- in the event that $D$ is not responsive for a defined time interval, $X$ sends periodic / (triggered) updates to its neighbors about $D$ by setting $c = INFINIT$ and $s_{new} = s + 1$

As the proposed protocol is implemented based on the distance vector algorithm [10], the DVRP module is implemented based on the complements and limitations of the algorithm. However, some techniques such as triggered updates hold-down timers, packet buffering and encapsulated update messages are used to address certain limitations inherited from the distance vector routing algorithm [10].

## 3.1 Class Overview

The *ns3::DVRP::RoutingProtocol* class is implemented to the ns-3 by extending the abstract base class *ns3::Ipv4L4Protocol*. The *ns3::DVRP::PacketHeader* and *ns3::DVRP::Hello* classes are extended from *ns3::Header*. The *ns3::DVRP::NeighborTableEntry* class is declared to store neighbor information and the *ns3::DVRP::NeighborTable* class is declared to store all neighbor records in a table. Similarly, the *ns3::DVRP::RoutingTableEntry* class is designed to store route information, while the *ns3::DVRP::RoutingTable* is declared to create routing tables. In addition, the *ns3::DVRP::QueueEntry* class was introduced to store packets and the *ns3::DVRP::PacketQueue* class was introduced to manage all stored queue entries. The *ns3::DVRP::RoutingProtocol* class is the main class that combines all the aforementioned classes and maintains route and neighbor management functions.

The DVRP module can be configured to a simulation topology using the same method that ns-3 provided to configure list routing protocols [5, 18, 20, 25]. As an example, first create an object of *DVRPHelper* and attach it to the *ns3::Ipv4ListRoutingHelper*. The *ListRoutingHelper* then must be attached to the *ns3::InternetStackHelper* to initialize the DVRP module.

## 3.2 Neighbor Management

The structure of the DVRP neighbor module consists of two main classes: the *NeighborTable* and *NeighborTableEntry* classes. The *NeighborTable* class is a collection of *NeighborTableEntry* class records.

### 3.2.1 Neighbor Table

As previously stated, the *NeighborTableEntry* contains neighbor information such as the IP address of the neighbor, local router's interface at which the neighbor can be reached, local forwarding socket, and last updated time. As depicted in Figure 3, neighbor records are mapped to the *NeighborTable*. The *NeighborTable* class has methods to add, remove, update, and print neighbor records.

**Table 1: DVRP Attributes and default Values**

| Attribute | Default Value | Summery |
|---|---|---|
| PeriodicUpdateInterval | 20s | Time between two periodic updates. |
| KeepAliveInterval | 30s | Time between two keep alive messages. |
| SettlingTime | 30s | Minimum time to settle a route record. |
| HoldTimes | 90s | Maximum time that a route record can wait without and updated. |
| MaxNeighborTime | 60s | Maximum time that a neighbor record can wait without an update. |
| MaxQueueLen | 1000 | Maximum number of packets that can be allowed to store in the queue. |
| MaxQueueTime | 45s | Maximum time that a packet can be in the Queue before it is discarded. |
| EnableBuffering | True | Enable packet buffering if no route found to its destination. |

```
for each KAM do
    check for existing records;
    if !record then
        add new record;
    else
        update the life-time of matched record;
    end
    for each neighbor record do
        if lifetime >MaxNeighborTime then
            delete the record;
            delete route records refer deleted neighbor;
        end
        send triggered update;
    end
end
```

**Algorithm 1:** Neighbor Management

| Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|
| Packet Type | < Not use > | | |
| Source Address | | | |

(a) DVRP Hello Packet Header

| Byte 1 | Byte 2 | Byte 3 | Byte 4 |
|---|---|---|---|
| Packet Type | | Index | Hop Count |
| Packet Sequence Number | | | |
| Source Address | | | |
| Destination Address | | | |

(b) DVRP Update Packet Header

**Figure 2: Routing Packet Headers**

### 3.2.2 *Processing Neighbor Advertisement Messages*

At the router initiation time, when the *ns3::NotifyInterfaceUp* callback is triggered, all local interfaces are added to the permanent routing table. Consequently, the router broadcasts hello packets, among its local interfaces, using the packet structure given in the Figure 2(a). After sending the first hello packet, the neighbor module is programmed to broadcast keep-alive messages in each *KeepAliveIntrerval*. This method has been designed to maintain consistency among routers. The same packet structure given in Figure 2(a) is used to send keep-alive messages.

As soon as a router receives a hello packet, the router processes the packet using the neighbor management algorithm (Algorithm 1). Initially, the router checks the received neighbor record for a match among existing neighbor records. If no match is found, the record is added to the neighbor table as a new neighbor. Otherwise, the router updates the last updated time of the matching record. In addition, the router checks for outdated neighbors using (1).

$$LifeTime = currentTime - lastUpdatedTime \tag{1}$$

In the event *LifeTime* exceeds *MaxNeighbourTime*:, the corresponding neighbor record is first removed from the neighbor table. Then, route records that refer the removed neighbor as the forwarding gateway are also removed from both advertisement and permanent routing tables. Finally, the router sends a triggered update message among remaining neighbors regarding the modified routing entries.

| Neighbor Address | Forwarding Interface | Socket Details | Life Time |
|---|---|---|---|

**Figure 3: Neighbor Table**

| Destination | Sequence Number | Network address | Next Hop | Forwarding Interface | Hop Count | Changed | Life Time |
|---|---|---|---|---|---|---|---|

**Figure 4: Main Routing Table**

| IP Header | UDP Header | DVRP Route message | DVRP Route message | DVRP Route message | ... |
|---|---|---|---|---|---|

**Figure 5: Route Message Encapsulation**

## 3.3 Route Management

DVRP module is a UDP-based protocol and uses UDP port number 272 for route updates. The DVRP module maintains two types of routing tables: permanent and advertisement. In addition, the DVRP module uses two types of route updates messages: periodic and triggered. Routers broadcast periodic update messages in every *PeriodicUpdateInterval* time. The triggered updates are particularly introduced to overcome the slow convergence limitation inherited by the distance vector algorithm [10].

Triggered update messages are not currently designed to send selective route records. Because triggered updates also broadcast entire routing tables, triggered updates may cause excessive loads on network links. Therefore, special handling is required to limit the network congestion caused by triggered updates. To avoid such loads, random intervals between 1 and 5 s are introduced as a provi-

```
purge routing table;
ROUTES <- collect routes and merge with purged records;
if !ROUTES then
    return;
end
for each neighbor in neighbor-table do
    create a empty packet;
    maxRum = calculate number of records based on MTU;
    for each record in ROUTES do
        if gateway == "0.0.0.0" then
            set the dest. address as this node;
            increment the Seq. no by 2;
            increment the hop count by 1;
            set all other header fields;
            update the routing table for new Seq. number;
            add the header to the packet;
        else
            set the dest. address according to the route record;
            if (Seq. % 2) == 0 then
                increment Seq. number by 2;
                increment the hop count by 1;
            else
                increment Seq. number by 1;
                set hop count for infinity;
            end
            set all other header fields;
            add the header to the packet;
        end
        if number of records == maxRum then
            send the update packet to the neighbor;
        end
    end
    if more records to send then
        send the update packet to the neighbor;
    end
end
```

**Algorithm 2:** Update Message Generation

sion. Therefore, after a triggered update is broadcasted, the router must wait 1 to 5 s before it sends the next update.

### *3.3.1 Routing Table*

previously stated, DVRP maintains mainly two routing tables to store permanent (stable) routes and recently received routes. As shown in Figure 4, a DVRP module uses a host-address-based routing table. Therefore, every route entry stores the following attributes about the route: a destination IP address, last updated sequence number, network address of the next hop, IP address of the next hop, hop count for the destination, forwarding interface address, and the last updated time. Further, a Boolean value is used to specify whether the route entry can be changed. In addition, the DVRP module uses a sequence number for every route record, while the sequence number is used for two purposes. The first purpose is to maintain route consistency. In every route update, sequence numbers of route records increment by one or two (to produce an odd or even value for the sequence number). The second purpose is to indicate non-responsive records. Non-responsive and responsive records have odd and even sequence number values, respectively.

Newly received routes must wait at the advertisement routing table until the *SettingTime* expires to determine whether the route is stable. The route record is then moved to the permanent routing table. A router does not send any advertisement about new route records before those records are moved to the permanent routing table. Stable routes are identified by the sequence number and hop count. That process is explained in Section 3.3.3.

### *3.3.2 Route Advertisements*

The route update packet is designed to have multiple route records added to it. A record is added to the packet using a 13-byte route message, the format for which is given in Figure 2(b). As shown, a 32-bit wide sequence number is used in the DVRP to incorporate the DVRP in large-scale networks. The hop count is limited to one byte, because the distance between two end hosts is assumed to be not more than 255 hops.

When a particular router generates a route update packet, as shown in Figure 5, the router encapsulates all update records into a single update packet based on the maximum transfer unit (MTU) of a particular link. If a route update packet exceeds the MTU of a particular link, the route update packet is split and broadcast among neighbors.

$$LifeTime > HoldTimes * PeriodicUpdateInterval \qquad (2)$$

At the route update generation time, as given in Algorithm 2, the DVRP module first creates an empty route update packet. The DVRP module then purges the routing table for any outdated entries. All outdated route records are removed from the routing table based on (2). Consequently, the removed route records are added to the update packet by setting the sequence number to an odd value and the cost to infinity. The DVRP module specially handles the route records of the router's local interfaces. First, all local interface route records are added to the update packet by incrementing the sequence number by two and the hop count by one. Thenceforth, to maintain an up-to-date sequence numbers for local interfaces, the DVRP module updates the sequence number of the local route records in the permanent routing table.

Finally, after adding the removed and local interface route records to the update packet, the DVRP module adds remaining routing records to the update packet .The DVRP module then broadcasts the update packet among all neighbors.

### *3.3.3 Processing Update Packets*

As the DVRP protocol broadcasts route updates, it is worthwhile to confirm whether the update packet is from one of the router's own interfaces. Therefore, when the DVRP module receives a route update packet, it checks the update packet for piggybacking. If the received packet is piggybacked, the packet is discarded. Otherwise, the packet is processed as given in Algorithm 3.

Three major factors are involved in the update packet processing method: the destination IP address, sequence number and hop count. The destination IP address of the extracted message is first checked. If the destination is the same as the routers's IP address, the message is discarded. If the destination is not listed in both routing tables, the message is verified for a new route with a valid sequence number. If the route is a new route, the route is added to the permanent routing table and broadcasts a triggered update among its neighbors. Otherwise, as the router has a route record for the destination IP address, the update message's sequence number will be checked. If the sequence number is an odd value and the message is intended for an unreachable host, certain route records are removed from both routing and advertisement tables and an immediate update is sent to neighbors. However, if the sequence number is even value, one of the following scenarios is used to update the routing table.

```
if piggybacked packet then
    discard packet and return;
end
for each RUM in the packet do
    if (Seq. % 2) != 0 then
        delete routes in routing table;
        delete routes in advertisement table;
        send a triggered update;
    end
    else
        ROUTE <- find(destination);
        calc HopCount;
        if !ROUTE then
            add a record to routing table;
            add a record to advertisement table;
        else
            if Seq. >= ROUTE'Seq. then
                if HopCount <= ROUTE'HopCount then
                    if different Gateway then
                        delete routes in both tables;
                        add route to adv. table;
                        schedule settling event;
                    else
                        update HopCount;
                        update the Seq.;
                    end
                else if HopCount >ROUTE'HopCount &&
                <16 then
                    if different Gateway then
                        update HopCount;
                        update the Seq.;
                    else
                        discard the message;
                    end
                end
            else
                discard the message;
            end
        end
    end
end
```

**Algorithm 3:** Route Processing

- If no route presents in the routing table, add new routes to both routing and advertisement routing tables.

- If the received sequence number is greater than or equal to the existing sequence number, then (3) is used to calculate the shortest path and to update the routing record in the advertisement table.

- If the hop count of the received record is less than or equal to the existing hop count, as given in Algorithm 3, update that particular record of the advertisement routing table based on the gateway address. If the sender address is different compared to the gateway address, delete the existing route records and add new route record to the advertisement table. An event will then be scheduled to expire after *SettingTime*. At the time the scheduled event triggers, the route record will be moved from the advertisement routing table to the permanent routing table. In particular, the hop count, sequence number, and last updated time of the route record will be updated.

- If the sender address is same as the gateway address, particularly hop count and the sequence number get updated according the route update message. In addition, the last updated time will also be updated.

- If the hop count is greater than the route's hop count and is lesser than sixteen, then hop count and the hop count will be updated if the gateway is same as the sender. Otherwise, the message will be discarded

- If the received sequence number is shorter than the existing sequence number, the message will be discarded, because the node has a most updated record than does the received update.

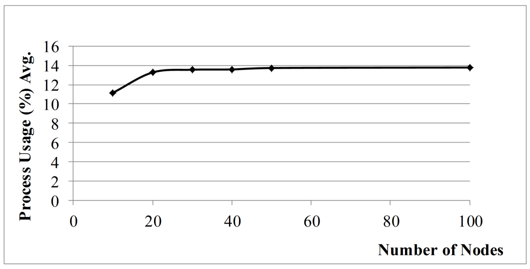

**Figure 6: Average Processor Consumption**

$$C^X(Y,Z) = C(X,Z) + min_w\{C^Z(Y,w)\} \tag{3}$$

$C^X(Y,Z)$ : distance between $X$ to $Y$ via $Z$
$C(X,Z)$ : distance between $X$ to $Z$, where $Z$ is the neighbor of $X$
$min_w\{D^Z(Y,w)\}$ : shortest hop path from $Z$ to $Y$

### 3.4 Packet Buffering

Per-router packet buffering is introduced in the DVRP module as an additional function. If a router does not have a route through which to forward a packet, instead of the packet being dropped, it will be buffered in the local packet buffer. The packet buffering feature is enabled by default to store 1000 packets in each router. Administrators can control the packet buffer option by controlling the *EnableBuffering*, *MaxQueueLen* and *MaxQueueTime* control variables given in Table 1.

The *PacketQueue* class was implemented to store packets using the *QueueEntry* class. The *QueueEntry* class stores following: the packet, packet's source IP address, packet's destination IP address, and the maximum permitted time to remain in the packet buffer. Packets will be buffered in the packet buffer until: a router receives a route to forward the packet, or the the maximum permitted time to remain in the packet buffer has expired.

## 4. EVALUATION AND DISCUSSION

The ns-3.18 version was used to evaluate the DVRP protocol implementation. To evaluate the functionalities of the implemented routing protocol, we implemented six mesh topologies containing 10, 20, 30, 40, 50, and 100 routers. Because the DVRP module is an Interior Gateway Protocol (IGP) routing protocol, we assumed that a maximum of 100 routers is sufficient for a network topology. The network links of those topologies were randomly created. A

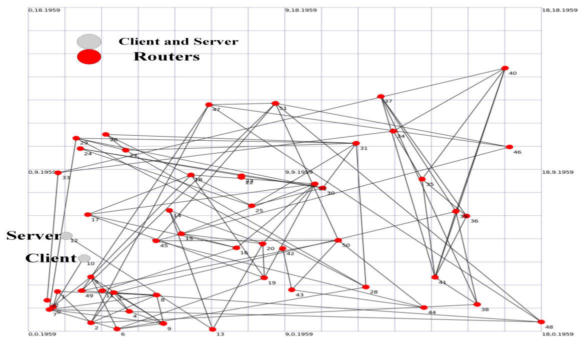

**Figure 7: Sample ns-3 Topology Consists of 100 Nodes**

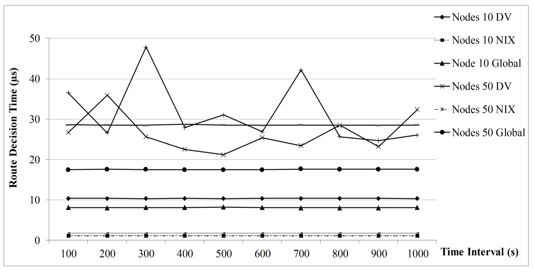

**Figure 8: Route Decision Making Time (Avg.)**

sample topology (of 100 routers) is given in Figure 7. The proposed DVRP protocol is compared with the ns-3 NIX-vector and ns-3 global routing protocols using the six topologies. Note: due to the fact that [5], [18] and [1] are wireless routing protocols, the proposed protocol is not compared with those routing protocols.

The topologies were implemented on a computer with an Intel i7 3.4 GHz processor having 8 GB of memory. The "sar" [12] Unix command was used to measure the process consumption. The obtained process usage result is plotted in Figure 6. According to the Figure 6, the topology consists of 100 routers consumed an average of 13.08% of the processor. Therefore, the DVRP module was able to utilize the processor usage to a reasonable degree.

## 4.1 Topology Development

The simulation topologies were configured using peer-to-peer links of 100 Mbps and 2 ms of delay. As shown in Figure 7, we used two links, 10 Mbps and 1 ms delay, to connect UDP echo client and UDP echo server to the network topology. We programed the client to send packets with 500 bytes payload at random intervals. The server was programmed to reply when the packet arrived. The DVRP, ns-3 NIX-vector, and ns-3 global routing protocols were configured to the aforementioned topologies. Each simulation scenario was simulated five times and the average results were used for the evaluation.

## 4.2 Results and Discussion

This section discusses the obtained results in detail. Tests were conducted to evaluate the DVRP for the route decision time (RDT), packet delivery ratio (PDR), round trip time (RTT), and routing overhead (RO).

### *4.2.1 Route Decision Time (RDT)*

The RDT value is the average time used by a router to discover

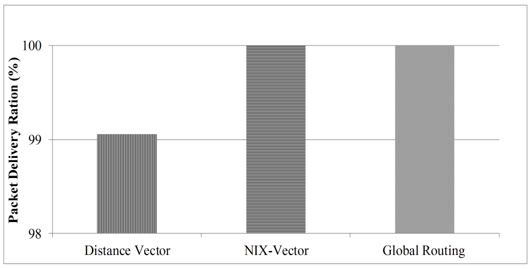

**Figure 9: Packet Delivery Ratio (Avg.)**

a route from the routing table to a packet. For this experiment, we used topologies that consist of 10, 50, and 100 routers. The measured RDT values for the routing protocols are given in Figure 8. As depicted, the NIX-vector routing protocol consumes an average of 0.1$\mu$s for all three topologies. This is because, as given in [19], the routing path for a particular data packet is determined during packet generation. Therefore, routers do not consume time to determine routes for incoming packets, so route decision time is extremely low compared to that in the other two routing protocols.

Both DVRP and ns-3 global routing protocols display increments of the RDT value proportional to the number of routers. The reason is that, latency of finding a route record is depending on size of the routing table and the size of the routing table is proportional to number of nodes in a network. However, according to Figure 8, the ns-3 global routing protocol consumes a steady time to determine a route. As an example, 28$\mu$s are consumed for the 100 routers topology. As explained in [20], this happens because when a data packet arrives, the ns-3 global routing protocol returns the first record of the routing table that matches the destination address of the received packet. Therefore, time consumption of finding a route in the routing table is stable. But, given the fact that the ns-3 global routing protocol maintains a routing table, the RDT value is high compared to that of NIX-vector routing.

Similarly, as Figure 8 shows, the RTD value of the DVRP also increases based on the number of routers in a topology. Further, as depicted in Figure 8, the RDT values also fluctuate according to the time. This is because the DVRP protocol must allocate resources to process both route update messages and incoming data packets. As an example, the routing table must be shared between the route update and packet forwarding modules. Therefore, as Figure 8 reveals, DVRP protocol displays an average of 10-20 $\mu$s increments of the RDT value compared to the ns-3 global routing.

### *4.2.2 Packet Delivery Ratio (PDR)*

The PDR value is calculated at the UDP echo client by dividing the number of received reply packets by the number of sent packets. According to the Figure 9, both NIX-vector and global routing protocols display 100% PDR. This is because these two routing protocols do not use any route update messages. Thus, the network is neither occupied nor congested. The DVRP protocol is programmed to send periodic and triggered updates as well as hello packets to maintain the topology consistency and adapt to topological changes. Route update packets occupy both the router and network to a certain percentage depending on the size of the update packet. Therefore, the proposed protocol displays 99.05% of the packet delivery ratio, which represents 0.95% packet loss compared to that of existing ns-3 routing protocols.

### *4.2.3 Round Trip Time Delay (RTT)*

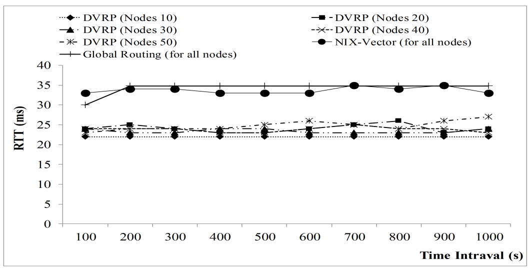

**Figure 10: RTT measurement at the client (Avg.)**

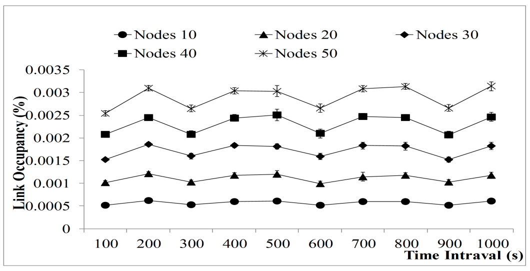

**Figure 11: Routing Overhead(Avg.)**

The third experiment in this study involved measuring the RTT values of reply packets at the UDP echo client. According to Figure 10, all topologies configured with both NIX-vector and global routing protocols result in an average of 35 ms of RTT. Topologies configured with the DVRP protocol resulted in an average of 25ms of the RTT value. This means that the DVRP protocol is 10 s faster than the existing ns-3 routing protocols in terms of packet delivery.

The NIX-Vector routing protocol has a comparatively high RTT value because the routing protocol analyzes data packets in each router to determine the next forwarding router. Therefore, a packet has considerable time in a router until the packet is forwarded to the next hop. In addition, the RTT value of the global routing protocol is high. According to [20], this is because the global routing module returns the first route of the routing table. Usually, the returned route is not the shortest path. Therefore, packets may travel in long distance paths.

According to the Figure 10, DVRP displays fluctuations of the RTT value proportional to the number of routers in the topology. One reason for this is that the DVRP module stores destination-based instead of network-based route records. Therefore, the size of the routing table is proportional to the number of routers in the networks. Another reason is that the number of update messages and update packet size is proportional to the size of the routing table. The route update messages are directly affect network congestion. Data packets may thus travel through congested routers or links. As a result, RTT values fluctuate based on the number of nodes in the network and because of network congestion.

In the next experiment, some routers were deactivated in the middle of the simulation. In addition, the behavior of the routing protocols was observed. In the case of NIX-vector routing, the packets were not delivered if the broken link was in the pre calculated routing path of a data packet. Similarly, for the global routing, if the deactivated router was referred by a route record, the packets were not delivered. However, DVRP module behavior was completely different compared to the NIX-Vector and global routing modules.

The DVRP module used keep-alive messages to identify the disabled router. After a router detects the deactivated router, it removes the deactivated router from its routing table and advertises among its neighbors. The entire topology then recalculates the shortest path according to the new topology. And packets are routed using the newly calculated shortest path. In addition, routers buffer the packets that cannot be delivered (because of the unresponsive router) until routers receive new routes.

#### 4.2.4 *Routing Overhead (RO)*

The fourth experiment was conducted to measure the link occupancy ratios of routing protocols. As the NIX-vector and the ns-3 global routing protocols do not use update messages to exchange route information, the comparison was not successful. However, the RO value of the DVRP is provided and discussed in this section.

The RO value is calculated dividing the number of bytes used for the route management by the bandwidth of the link. The results are plotted in Figure 11. It shows that the DVRP protocol consumed an average of 0.003% of a 100 Mbps link in a 50-router network topology. In addition, the DVRP module displays 0.005% increments of the RO value proportional to the number of nodes in a topology. As previously stated, this is because the routing table size and number of update messages increase based on the number of nodes on a topology. However, the number of update packets did not drastically increase proportional to the number of nodes in a topology. Because the DVRP module encapsulates more than one route update message in a single update packet in order to maintain network utilization.

## 5. CONCLUSION AND FUTURE STUDIES

In this study, we introduced and implemented an IPv4 table-driven wired routing protocol for an ns-3 simulator. The protocol was designed as a distance vector routing protocol (DVRP). This paper provides a detailed explanation of the design, classes, and test results. The test results showed that the DVRP could deliver a packet on an average of 10-13 ms faster compared to existing ns-3 IPv4 routing protocols. Furthermore, the DVRP module utilized only 0.003% of links in its route management. However, test results showed that the DVRP is 0.95% low in packet delivery because of the limitations inherited from the distance vector routing algorithm.

The DVRP protocol was able to address successfully the limitations of the existing IPv4 routing protocols of the ns-3. Specifically, it used route update messages to populate the routing table and adapt to topological changes. In addition, the DVRP was enhanced to buffer undelivered packets for a pre-programmable time instead of dropping those packets. However, due to the fact that [5], [18] and [1] are wireless routing protocols, the DVRP protocol is not evaluated based on those protocols. Further the DVRP is not evaluated based on the RIPng protocol [11] because it a IPv6 based routing protocol.

The source code of the current version of DVRP can be found in [29]. In the future , the DVRP module will be developed to support both VLSM and CIDR.

## Acknowledgments

This work was partially supported by the funds of SECOM Science and Technology Foundation, and by MEXT/JSPS KAKENHI Grant (B) Number 24360230 and 25280033. It is also supported by the Japanese Government (Monbukagakusho: MEXT) Scholarship for Research Students, and the Keio University KLL Ph.D. Program Research Grant.

## 6. REFERENCES

[1] Bikov, E., and Boyko, P. Direct execution of olsr manet routing daemon in ns-3. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques (SIMUTools '11)* (2011), pp. 454–461.

[2] Brakmo, L. S., and Peterson, L. L. An evaluation of the network simulators in large-scale distributed simulations. In *In Proceedings of the 1996 ACM SIGMETRICS international conference on Measurement and modeling of computer systems (SIGMETRICS '96)* (1996), pp. 80–90.

[3] Carneiro, G. J. Ns-3 network simulator 3. `https://www.nsnam.org/tutorials/NS-3-LABMEETING-1.pdf`, Accessed: 6/2015.

[4] Cheng, C., Riley, R., Kumar, S. P. R., and Garcia-Luna-Aceves, J. J. A loop-free extended bellman-ford routing protocol without bouncing effect. *SIGCOMM Comput. Commun. Rev. 19*, 4 (Aug. 1989), 224–236.

[5] Cheng, Y., Cetinkaya, E. K., and Sterbenz, J. P. G. Dynamic source routing *dsr* protocol implementation in ns-3. In *In Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques SIMUTOOLS'*12 (2012), pp. 367–374.

[6] Ciraci, S., and Akyol, B. Experiences with network simulation. In *In Proceedings of the first international workshop on High performance computing, networking and analytics for the power grid (HiPCNA-PG '11)* (2011), pp. 59–66.

[7] Dally, W., and Towles, B. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[8] DARPA. The network simulator: ns-2, howpublished =.

[9] Das, S., R., C., Yan, J., and R., S. Comparative performance evaluation of routing protocols for mobile, ad hoc networks. In *In Proceedings of the 7th International Conference on Computer Communications and Networks* (1998), pp. 153–161.

[10] Dordal, P. L. An introduction to computer networks. `http://intronetworks.cs.luc.edu/1/html/routing.html`, Accessed: 6/2015.

[11] G. Malkin, R. M. Ripng for ipv6. `http://datatracker.ietf.org/doc/rfc2080/`, Accessedd: 6/2015.

[12] Godard, S. sar(1) - linux man page. `http://linux.die.net/man/1/sar`, Accessed: 6/2015.

[13] Heap, G. T., and Maynes, L. *CCNA Practical Studies*. Cisco Press, 2002.

[14] Hedrick, C. Routing information protocol. `http://tools.ietf.org/html/rfc1058`, Accessedd: 6/2015.

[15] Ikeda, M., Kulla, E., Barolli, L., Takizawa, M., and Miho, R. Performance evaluation of wireless mobile ad-hoc network via ns-3 simulator. In *2011 14th International Conference on Network-Based Information Systems (NBiS)* (2011), pp. 135–141.

[16] KIM, J. Shortest path routing protocol for ns3. `http://www2.engr.arizona.edu/~junseok/shortest_path.html`, Accessed: 6/2015.

[17] Kumar, A., Rao, S., and Goswami, D. Ns3 simulator for a study of data center networks. In *2013 IEEE 12th International Symposium on Parallel and Distributed Computing (ISPDC)* (2013), pp. 224–231.

[18] Narra, H., Cheng, Y., Cetinkaya, E. K., Rohrer, J. P., and Sterbenz, J. P. G. Destination-sequenced distance vector (dsdv) routing protocol implementation in ns-3. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques (SIMUTools '11)* (2011), pp. 439–446.

[19] ns 3-dev. ns-3 nix-vector routing. `http://www.nsnam.org/doxygen-release/group__nix-vector-routing.html`, Accessed: 6/2015.

[20] ns 3-dev. ns-3 routing overview. `https://www.nsnam.org/docs/release/3.10/manual/html/routing.html`, Accessed: 6/2015.

[21] ns 3-dev. ns-3 simulator. `http://www.nsnam.org`, Accessed: 6/2015.

[22] Nygren, E., Sitaraman, R. K., and Sun, J. The akamai network: A platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev. 44*, 3 (2010), 2–19.

[23] Pecorella, T. Ripng. `http://www.nsnam.org/doxygen-release/group__ripng.html#details`, Accessed: 6/2015.

[24] Spaho, E., Ikeda, M., Barolli, L., Xhafa, F., Younas, M., and Takizawa, M. Performance of olsr and dsdv protocols in a vanet scenario: Evaluation using cavenet and ns3. In *Proceedings of the 2012 Seventh International Conference on Broadband, Wireless Computing, Communication and Applications* (2012), BWCCA '12, pp. 108–113.

[25] Suresh, P. L., and Merz, R. ns-3-click: click modular router integration for ns-3. In *In Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques SIMUTools'*11 (2011), pp. 423–430.

[26] Swenson, B. P., and Riley, G. F. Simulating large topologies in ns-3 using brite and cuda driven global routing. In *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques (SimuTools '13)* (2013), pp. 159–166.

[27] Wijekoon, J., Harahap, E., Ishida, S., Tennekoon, R., and Nishi, H. Router-based content-aware data redirection for future cdn systems. *International Journal of Computer Network and Information Security (IJCNIS) 6*, 7 (2014), 1–10.

[28] Wijekoon, J., Tennekoon, R., Harahap, E., and Nishi, H. Service-oriented router module implementation on ns-3. In *SIMUTOOLS 2014: The 7th International ICST Conference on Simulation Tools and Techniques* (March, 2014).

[29] Wijekoon, J., Tennekoon, R., and Nishi, H. ns3 dvrp implementation. `https://github.com/westlab/ns3dvrp`, Accessed: 6/2015.