

Exploiting Local IPv4-only Access Networks to Deliver IPv6 Service to End-users

Ala Hamarshah and Marnix Goossens

Abstract—This paper introduces a new IPv6 deployment protocol called E4Deliver6, which stands for Exploiting Local IPv4-only Access Networks to Deliver IPv6 Service to End-users. The protocol intends to help the Internet Service Providers (ISPs) to rapidly start providing IPv6 service to the end-users. E4Deliver6 tunnels the IPv6 traffic into IPv4 traffic to transit the IPv4-only access network infrastructure. The new protocol offers IPv6 service alongside IPv4 service to the end-users, economical solution, and requires simple and automatic configurations at both end-users' hosts and ISP sides at the time of setup. E4Deliver6 connected hosts will be able to communicate with other IPv6-only servers outside their local IPv4-only access network. The performance analysis of E4Deliver6 showed that the performance parameters (e.g. Latency and Throughput) are acceptable in comparison to other stateless protocols (e.g. 6rd) and it is much better than other stateful protocols (e.g. BDMS) performance parameters.

Keywords— Local IPv4-only access networks, IPv6 deployment, IPv6 host, IPv6-in-IPv4 tunnels, IPv4 host, ISP, Latency, Throughput.

I. INTRODUCTION

TRADITIONALLY, IP addresses are used to uniquely identify network connected devices. The lack and shortage in IPv4 [1] available addresses lead to develop a new addressing scheme (IPv6 [2]) by Internet Engineering Task Force (IETF). The new IPv6 protocol has various advantages over existing IPv4 protocol. For example, it offers 128-bit addresses, better security (IPsec [3] [4] [5]), efficiency, QoS, mobility, etc.

Unfortunately, the two protocols are incompatible with each other, and changing the IP address structure effects on the whole network stack. Since the socket layer is the part of the network stack, it will be affected in this change as well. Most network applications are built to use specific type of socket

layer functions, and changing the IP layer will lead to change all these functions in which they allow running applications to communicate over current host's network connectivity. In such a way, all applications that are using these old socket Application Programming Interface (API) functions need to be rebuilt in order to make them capable to work over new protocol. DAC [6] overcomes the incompatibility problems that might exist between network applications and host's connectivity. It allows any application type (IPv4/IPv6) to work over any communication protocol (IPv4/IPv6) without changing these applications in addressing capabilities.

However, there are many other areas of incompatibilities between IPv4 and IPv6 that arise while deploying IPv6 service. The task of adopting IPv6 is not an easy task and it is quit complex. Transition from IPv4 to IPv6 should appear in steps starting from few IPv6 nodes, and the number should gradually increase over the time until the whole network becomes IPv6 [7]. Therefore, the coexistence between IPv4 and IPv6 will remain for considerable amount of time. The IETF has proposed many transition mechanisms and specific types of IPv6 addresses to facilitate the communication between two nodes or networks operating on different IP architectures. Such mechanisms can be categorized into dual-stack, tunneling, and translation [8] [9] [10] [11] [12] [13].

This protocol uses tunneling technique to carry IPv6 traffic over local IPv4 access networks. Providing IPv6 connectivity to end-users requires changing/upgrading most network components on both sides (ISP and end-user sides). This protocol aims to provide IPv6 service to end-users with minimum change in the local IPv4 access network infrastructure (i.e. minimal possible cost). It does not require changing/upgrading network devices at the ISP side or Customer Edges (CEs). However it requires installing some network components at the ISP side, besides upgrading users' hosts by installing a particular application which is capable to encapsulate / de-encapsulate the outgoing/ingoing traffic over IPv4 local access networks, as well as to locate the tunnel endpoint (tunnel server). ISPs can continue providing IPv4 connectivity to their customers alongside IPv6 connectivity via E4Deliver6 protocol. With E4Deliver6, all parties use a common IPv6 prefix. This makes the proposed mechanism distinguishable from other IPv6 deployment mechanisms. Fig. 1 shows E4Deliver6 architecture deployed by an ISP.

Manuscript received May 22, 2011; Revised version received xxx. This work was funded by European Union, Erasmus Mundus External Cooperation Window (EMECW) programme under project number 141085-EM-1-2008-BE-ERAMUNDUS-ECW-L02, Belgium.

Ala Hamarshah, Vrije Universiteit Brussels, Faculty of Engineering, Department of Electronics and Informatics ETRO, Building K, Office No 4k222, Tel. +3226292930, Fax: +3226292883, Pleinlaan 2, 1050 Elsene, Brussels, Belgium (e-mail: ala.hamarshah@vub.ac.be).

Marnix Goossens, Vrije Universiteit Brussels, Faculty of Engineering, Department of Electronics and Informatics ETRO, Building K, Office No 4k220, Tel. +3226292930, Fax: +3226292883, Pleinlaan 2, 1050 Elsene, Brussels, Belgium (e-mail: marnix.goossens@vub.ac.be).

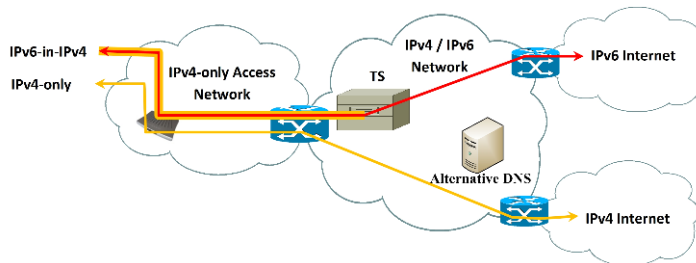


Fig. 1 E4Deliver6 protocol architecture

II. RELATED PROTOCOLS

A. IPv6 Rapid Deployment (6rd)

6rd mechanism [14] [15] is used by ISPs to facilitate deploying IPv6 service across IPv4 networks. Like 6to4 mechanism, it is used to transfer IPv6 packets over IPv4 networks. Unlike 6to4, 6rd mechanism only operates within the ISP. 6to4 uses fixed IPv6 prefix (i.e. 2002::/16), whereas in 6rd mechanism, each ISP uses its own IPv6 prefixes which can reflect good network management and quality of service to end-users. 6rd tunnels are created between CEs and 6rd border relay router.

E4Deliver6 protocol is similar in functionality to 6rd; however, in E4Deliver6 there will be no need to upgrade/change any of CEs. The end-users' hosts should be upgraded to work over E4Deliver6 protocol. The protocol allows IPv6-in-IPv4 tunnels to be created between E4Deliver6 based hosts and ISP tunnel server. Since there will be no need to change CEs at the customer side (minimal cost expected), this will encourage both parties (i.e. end-users and ISPs) to start adopting E4Deliver6 protocol and hence, moving toward deploying IPv6.

B. Bi-directional Mapping System

The Bi-Directional Mapping System (BDMS) [16] mechanism allows the communication between two different IP zones (from IPv4 to IPv6 and vice-versa) by translating the addresses and the packets for each communication session. The BDMS domain consists of DNS4 servers, DNS6 servers, DNS v4-v6 server, v4-v6 enabled gateway, IPv4-only hosts, and IPv6-only hosts. The mechanism mainly translates the IPv6 addresses into public IPv4 addresses from assigned IPv4 public address pool. Moreover, to allow the communication between IPv4 and IPv6 zones, it uses one of the protocol based translation mechanisms to translate the transmitted packets from one protocol to another. For example, when a host in an IPv6 network is willing to communicate with other server located in the IPv4 network, the BDMS mechanism will translate the private IPv4 into public address, then maps this public address with corresponding IPv6 address, and send the resolved IPv6 address back to the requesting IPv6 host. When the packet delivered at the v4-v6 enabled gateway, first of all, change the destination address into IPv4 address, then

translate the payload into IPv4, and finally, forward the IPv4 packet to its destination.

The drawbacks of this mechanism are: (1) both IP zones must be configured with the v4-v6 enabled gateway and with DNS46 modules. (2) it assigns public IPv4 addresses from address pool. However, this will not be in favor of serving the available IPv4 address space. (3) a lot of delay is generated by the DNS response time due to the propagation requests through different types of DNS servers to resolve the destination host's address. Moreover, additional overhead is generated in translating the IP headers of the transmitted packets from one protocol to another (see later).

III. ICMP ROUTER DISCOVERY AND SOLICITATION MESSAGES

In general, the ICMP router discovery messages are used by the host to discover the closed router(s), and they are used by routers to announce for their existence. These messages are categorized into ICMP router advertisement and solicitation messages. RFC 1256 [17] specifies the formats of ICMP messages. The following figures illustrate the format of the ICMP router solicitation and router advertisement messages exchanged between hosts and routers.

Type (10)	Code (0)	Checksum
Unused (sent as 0)		

Traditional solicitation message format:

Type (9)	Code (0)	Checksum
Number of addresses	Address ent. size (2)	Life time
Router address [1]		
Preference level [1]		
Router address [2]		
Preference level [2]		
.....		

Traditional advertisement message format:

After exchanging a certain number of router advertisements and/or solicitation messages, the local host can configure its routing table to be able to locate its default router for its outgoing traffic.

As we can see in the previous figures, the values of type field in each of the solicitation and advertisement messages are 10 and 9 respectively, and the value of code field is left 0 for both of them.

IV. PROTOCOL SPECIFICATIONS

E4Deliver6 specifies a protocol mechanism to deploy IPv6 service to hosts via existing service provider's (ISP) IPv4 network. The proposed protocol relies upon an algorithmic mapping between the IPv6 and IPv4 addresses that are assigned for use within the ISP network. The tunnel endpoints could be automatically determined by resolving the Tunnel Server (TS) domain name. This protocol would help in handling the hardest scenario among all the scenarios in [18] number 4 (IPv4 network to the IPv6 Internet), hence its operation will be stateful (see later).

A E4Deliver6 domain consists of customers' hosts and one or more TSs. Like 6to4 mechanism, the E4Deliver6 based host encapsulates the IPv6 packets and forwards them to follow the IPv4 routing topology within the ISP network. Thereafter, the TS will decapsulate the received packets and forward them to the IPv6 network. The TS is traversed only when any of the E4Deliver6 based hosts sends IPv6 packets or there are incoming IPv6 packets arriving from outside and destined to the ISP's E4Deliver6 domain.

The protocol aims to use the IPv4 infrastructure to deliver IPv6 connectivity alongside native IPv4 with a little change to IPv4 networking and operation as possible. It is important to note that the ISP network can continue deploying the IPv6 within their network while delivering the IPv6 service to the end-users' hosts.

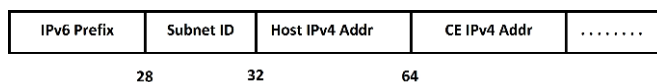
A. E4Deliver6 Host Based Architecture and Behavior

E4Deliver6 based hosts appear as native IPv6 hosts outside their local access networks. Such hosts must be able to automatically locate TS, create virtual IPv6 address, and (encapsulate/ decapsulate) the (outgoing/ingoing) traffic. E4Deliver6 hosts are configured with an application called Tunneler. The Tunneler module is responsible for the followings:

1) Creating Virtual IPv6 address:

IANA should reserve 28 bits well-known IPv6 prefix [19] belongs to this protocol. The virtual IPv6 address (i.e. not associated with IPv6 interface) for the host will be created by concatenating the E4Deliver6 prefix and the consecutive set of bits from the IPv4 address of the host and the IPv4 address of the customer edge (e.g. ADSL router).

The following figure shows the format of IPv6 address (Section 2.5.4 of [2]) with a E4Deliver6 prefix, host embedded IPv4 address, and customer edge embedded IPv4



address:

2) Locating Tunnel Endpoint:

The ICMP router discovery messages (i.e. router advertisements and router solicitations messages) are used by the host Tunneler to periodically check the availability and durability of TS at the ISP side. The host Tunneler starts by

resolving the domain name for the TS. Receiving a reply is an indication for the availability of E4Deliver6 service at the ISP side.

To check the durability of the service at the ISP side, the host Tunneler generates a unicast ICMP router solicitation message destined to TS directly. Receiving a reply from TS is an indication of service availability. In order to make E4Deliver6 ICMP messages distinguishable from traditional ICMP messages, a small change is applied by modifying the value of code field of ICMP messages from 0 to 1.

3) Encapsulation/Decapsulation:

All outgoing IPv6 traffic that generated by E4Deliver6 host will be encapsulated into IPv4 packets and forwarded over IPv4 infrastructure. The process of IPv4-in-IPv6 encapsulation goes as follows:

➤ IPv6 Header:

- Source Address: the virtual IPv6 address.
- Destination Address: the IPv6 address of the destination host.

➤ IPv4 Header:

- Source Address: the E4Deliver6 host IPv4 address (assigned by DHCPv4)
- Destination Address: the TS IPv4 Address.

The Tunneler upon receiving the encapsulated packets, it will decapsulate them, and forward the generated IPv6 packets into IPv6 stack.

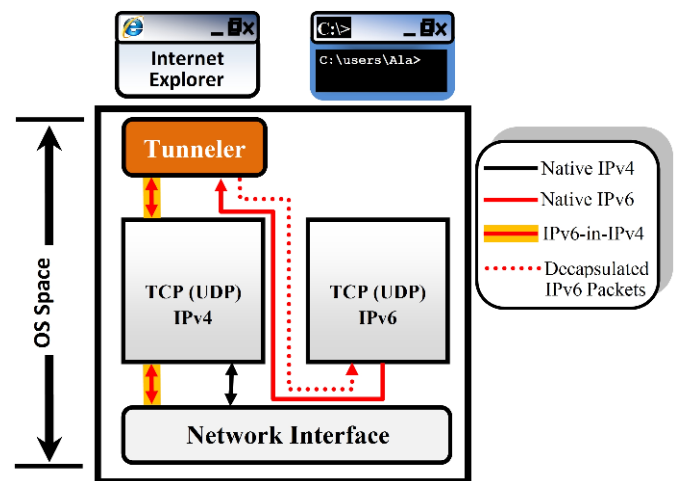


Fig. 2 E4Deliver6 host architecture

Forwarding manipulations and encapsulation techniques of IPv6-in-IPv4 are performed as described in [section 3.5 of [20]], with the same technique used by 6to4. Also the ICMPv4 errors are treated as described in [section 3.4 of [20]]. By default, the IPv6 traffic class field must be copied to the IPv4 Type of Service Field (ToS), or it could be overridden by

configuration. Fig. 2 shows E4Deliver6 host architecture.

B. ISP Network Configurations and Behavior

As mentioned earlier, ISPs can continue providing IPv4 connectivity to their customers alongside IPv6 connectivity via E4Deliver6 protocol. Some users will be upgraded to work over E4Deliver6 protocol and others will not. Thus, ISPs should be capable to provide different types of services (IPv4 and IPv6 services). E4Deliver6 ISP should be configured with the following components:

1) Alternative DNS:

Traditionally, each ISP has a local DNS (alternative DNS [21]) that used to resolve the domain names into IP addresses; hopefully this would be done locally. At the time of setup, the ISP network administrator should manually configure the local DNS and add a new record into the DNS records to denote for the TS. This allows the TS resolvable by other E4Deliver6 hosts, fortunately this happens only once at the setup time. The TS must have a unique domain name added to the local DNS; this domain name must be the same for all ISPs deploying E4Deliver6 protocol. The domain name must be chosen carefully so that it will not match any other domain name in the Internet (i.e. select a domain name with 256 characters). The ISP network administrator can assign any IP address for the TS domain name. Thus, E4Deliver6 based hosts can automatically build their tunnels after resolving the TS domain name.

2) TS:

TS is responsible for decapsulating the received IPv6-in-IPv4 packets and then forwarding the generated IPv6 packets to their destinations inside IPv6 Internet. Furthermore, encapsulating the received IPv6 packets into IPv6-in-IPv4 packets and then forwarding them to the E4Deliver6 host. TS always acts as a tunnel endpoint.

TS has at least one IPv4 interface and at least one IPv6 interface. Each TS has synthesized IPv6 address (IPv6 prefix + IPv4 address) assigned to it, and unicast IPv4 address.

The E4Deliver6 is a stateful operation, this means it maintains a conceptual dynamic data structure to record address information needed for the packet submission and reception. The table structure and functionality is similar to the one that used in stateful NAT64 mechanism [22].

The following table illustrates the TS mapping table structure:

SCR IPv4	TS IPv4	Host IPv4	CE IPv4	SCR Port	Mapped Port	SCR IPv6	DST IPv6
----------	---------	-----------	---------	----------	-------------	----------	----------

The TS dynamic data structure is a mapping table to record the addressing information of the active connections initiated from E4Deliver6 hosts and destined to IPv6-only Internet. TS is responsible for managing the mapping table (deleting or adding records). The mapping table should be supported by round-robin scheme to drop unused mappings for the longest

time. This would be sufficient for operational situations.

TS extracts the host IPv4 address, CE IPv4 address, and source port number. All these values can be found in the IPv4 header of the received packet.

When the outgoing packet is received at the TS, the TS performs the following actions:

- 1) Copy the source IPv4 address from the IPv4 header.
- 2) Strip off the IPv4 header.
- 3) Extract the E4Deliver6 based host IPv4 source and the CE IPv4 addresses from E4Deliver6 IPv6 virtual address.
- 4) Add TS IPv6 address as source address.
- 5) Map the source port number into corresponding mapped port number.
- 6) Record all previous information in the TS mapping table.
- 7) Forward the extracted IPv6 packet to the IPv6 Internet.

When the reply IPv6 packet is received at the TS device arriving from IPv6 host located in the IPv6 Internet (because TS IPv6 address starts with E4Deliver6 assigned prefix), TS performs the following actions:

- 1) Lookup in the mapping table to check if there was a previous mapping for the IPv6 destination and port number or not.
- 2) In case a mapping entry related to this packet is found, retrieve the related record immediately.
- 3) Use retrieved information to create E4Deliver6 IPv6 virtual address.
- 4) Configure IPv4 and IPv6 headers as follows:
 - **IPv4 Header:**
 - Source Address: TS IPv4 Address.
 - Destination Address: Closest router IPv4 address (it got this address when the outgoing packet is received at the TS device).
 - **IPv6 Header:**
 - Source Address: destination host IPv6 address.
 - Destination Address: E4Deliver6 virtual address.
- 5) Forward the packet to E4Deliver6 based host.

The IPv4 header is then added as it normally would be for any packet destined to E4Deliver6 based host. That is, the IPv4 destination address is the E4Deliver6 based host, and the source address is the TS IPv4 address. If the TS cannot find a record in the table, then it silently drops the packet. As explained earlier, both E4Deliver6 based host and TS use the same IPv6 prefix in generating E4Deliver6 virtual IPv6 addresses. TS is algorithmically configured to create E4Deliver6 IPv6 virtual address using IPv6 assigned prefix. Moreover, the same prefix is used when synthesizing the IPv6 address of the TS device. The TS configures the source address of the outgoing IPv6 packet header by changing the source address with its synthesized IPv6 address. Fig. 3 shows a flowchart to illustrate the functionality of TS.

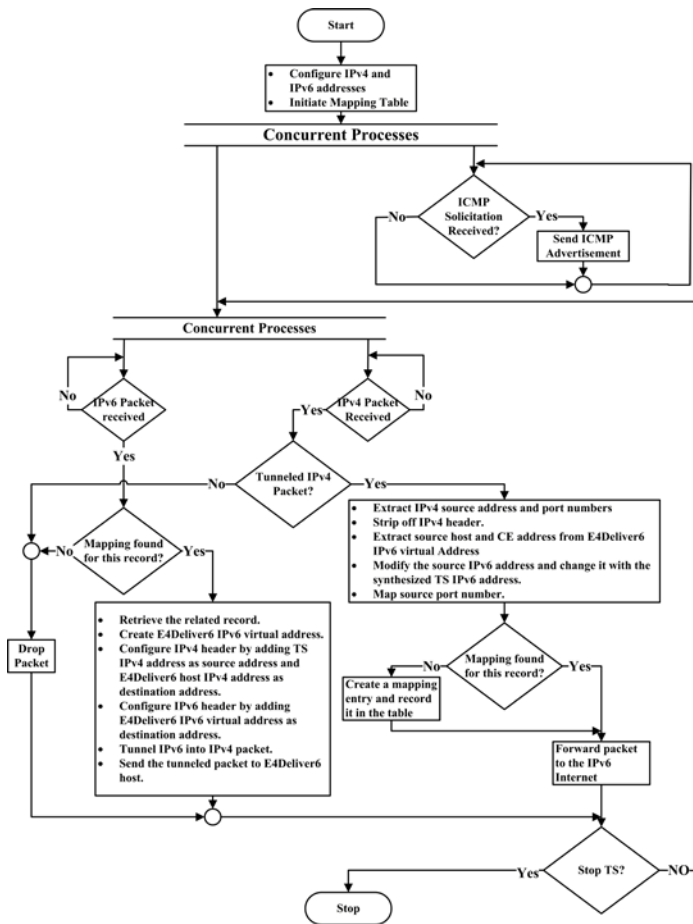


Fig. 3 TS functionality

V. PERFORMANCE ANALYSIS

The OMNET++ network simulation framework [23] was used to simulate 6rd, BDMS, and E4Deliver6 protocols. The goal of our simulation was to check the validity and to compare the performance of E4Deliver6 based networks with other related protocols in large scale settings. The performance is measured in terms of latency and throughput [24] parameters (see the next two subsections).

The simulator was implemented to simulate different scenarios. Each of these scenarios measured the performance parameters of a certain protocol type (e.g. E4Deliver6, 6rd, BDMS) with varying packet sizes (64, 128, 256, 512, 768, 1024 bytes) and varying number of nodes in the local access network (100, 300, 500, and 1000 nodes). The following scenarios are:

- 1) E4Deliver6 clients and IPv6 servers (E4Deliver6 network).
- 2) 6rd clients and IPv6 servers (6rd network).
- 3) IPv6 client clients and IPv4 servers through BDMS.

In addition to the protocol architecture that appears in Fig. 1, the simulator measures the latency and throughput parameters of 6rd client while communicating with IPv6 server. Furthermore, it measures the performance parameters of IPv6 client while communicating with IPv6 server through BDMS. Tests were conducted for each scenario by generating a traffic using TrafGen [25] tool. We used TrafGen tool to generate TCP traffic between clients and servers. The TrafGen tool has the ability to define the parameters for certain traffic flows (i.e. packet size, destination, ON and OFF in seconds, etc.) between client and server. For the sake of reliability and consistency, every test was repeated 50 times. The reported value at a specified packet size is the average of all recorded values.

Series of simulation results were performed in measuring the latency and throughput parameters at the above-mentioned packets' sizes and nodes. The E4Deliver6 based network performance parameters (latency and throughput) were compared to both 6rd and BDMS based networks performance parameters. In general, the results showed that E4Deliver6 parameters are roughly equal to 6rd network and it is much better than BDMS network performance parameters. However, E4Deliver6 performance tests showed a small lower performance in comparison to 6rd tests. We clarify this difference due to the overhead generated by the stateful operation of E4Deliver6 that allows it to frequently access to the mapping table at each outgoing/ingoing packet arrived at TS module.

A. Latency

The latency was measured as the time taken for a packet to be transmitted between E4Deliver6/6rd/ BDMS host and IPv6 server. Traditionally, IPv6 based networks produce higher latency values than IPv4 based networks. This happens due to the larger IPv6 header size (i.e. 40 bytes) in comparison to IPv4 header size (i.e. 20 bytes). In this simulation, the latency values were calculated to find the amount of overhead that each network type (i.e. E4Deliver6, 6rd, and BDMS) produces. Thus, we can indicate the QoS of each network type. The latency was calculated for each node individually as $Latency_{NODE}$, and then the $Latency_{NETWORK}$ can be computed as the average of all $Latency_{NODE}$ values in the network at a specified packet size and a certain number of nodes in the local access network. The equations Eq. (1) and Eq. (2) illustrate how $Latency_{NODE}$ and $Latency_{NETWORK}$ were calculated in all network types.

$$Latency_{NODE} = \frac{\sum_{i=1}^N (Td_i - Ts_i)}{N} \quad (1)$$

$$Latency_{NETWORK} = \frac{\sum_{j=1}^H Latency_{NODE}(j)}{H} \quad (2)$$

Where,

$Latency_{NODE}$: latency value for a certain node.

$Latency_{NETWORK}$: latency value for the whole network

T_d : is the time at destination host.

T_s : is the time at source host.

N : number of transmitted packets.

H : number of nodes in the network.

Figures (i.e. Fig. 4(A), Fig. 4(B), Fig. 4(C), and Fig. 4(D)) shows a comparative $Latency_{NETWORK}$ values for all protocols (e.g. BDMS, 6rd, and E4Deliver6) with varying number of nodes and packet sizes. Traditionally, increasing the packet sizes produces larger delay in delivering the packets, and hence increases the overhead. However, in all simulation scenarios, the packet size factor effects in two types of overheads. The first type is the overhead caused by network delays and the second type caused by network operations.

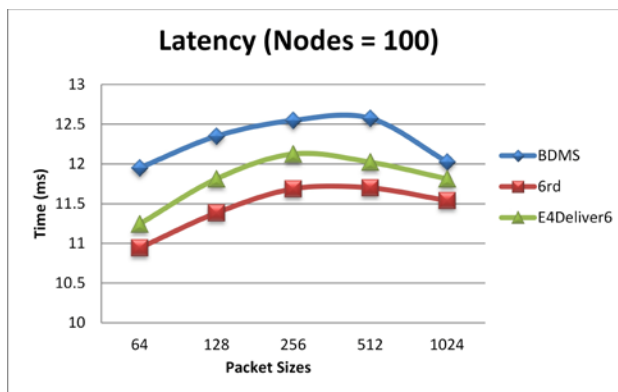


Fig. 4(A) Latency analysis – 100 nodes

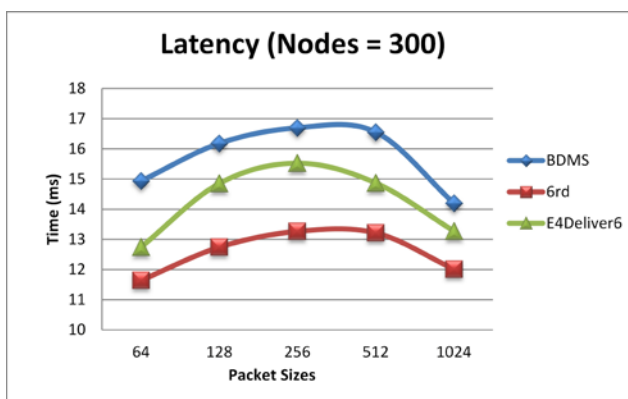


Fig. 4(B) Latency analysis – 300 nodes

In these simulation scenarios, the total overhead can be calculated as the overhead caused by the delay that was generated by transmitting larger packet sizes plus the overhead that was generated by encapsulation/decapsulation/translation the transmitted packets. The second type of overhead does not exist in native IPv4 and IPv6 based networks (i.e. does not deploy any IPv6

transition protocol). Therefore, larger packet sizes leads to

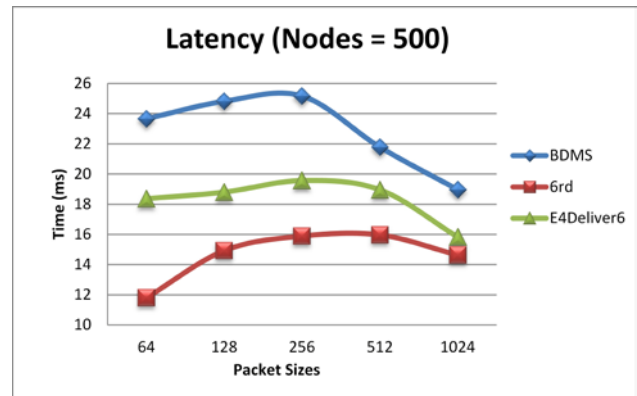


Fig. 4(C) Latency analysis – 500 nodes

increase the delay in these networks.

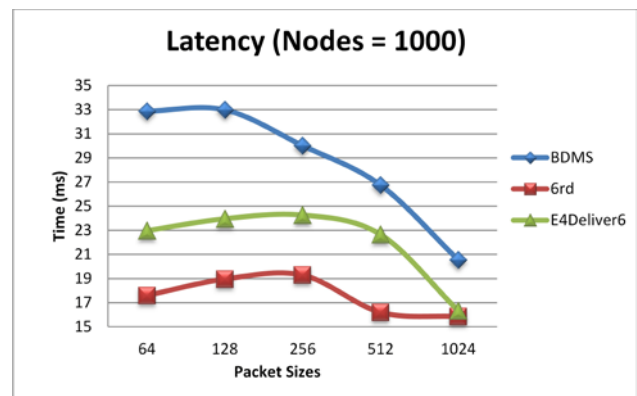


Fig. 4(D) Latency analysis – 1000 nodes

However, in other tunneling or translation based networks, larger packet sizes lead to decrease the delay in these networks. Referring to the simulation results of all conducted tests, the amount of delay that caused by tunneling, translation, or accessing mapping tables is much higher than the amount of delay that caused by transmitting large packet sizes. Therefore, the first type of overhead in these networks is very small in comparison to first type and it can be ignored.

It is important to highlight here that in the networks that use one of protocol tunneling or translation based techniques, transmitting larger packet sizes leads to decrease number of transmitting packets in the network. Thus, this will lead to decrease the delay that will be generated by encapsulating and decapsulating the packets, and hence decrease the amount of overhead in these networks. In all the figures from Fig. 4(A) to Fig. 4(D), the latency values were decreased by increasing the packet sizes except at the small packet sizes. Transmitting smaller packet sizes (i.e. smaller than 256 bytes) leads to transmit huge number of packets which need to be translated or tunneled, as well as, translates the addresses for each packet which produces additional overhead in these networks. BDMS achieved the highest latency values among other protocols due

to the frequent access to the DNS6, DNS4, DNS v4-v6 servers and translating the IP headers for every packet received. On the other hand, the latency values of 6rd and E4Deliver6 protocols are nearly equal. However, 6rd showed better latency values due to its stateless operation (i.e. does not maintain tables to record addressing information) in comparison to stateful operation (i.e. maintain tables to record addressing information) in both E4Deliver6 and BDMS protocols.

BDMS achieved the highest latency values especially when large number of nodes was used in the network. BDMS consumes very long time in resolving and translating addresses due to the distribution of all DNS servers between the networks. These DNS servers have to be referenced remotely by the translator of BDMS at every packet received. Although both BDMS and E4Deliver6 protocols are stateful operation, E4Deliver6 achieved better latency values in all conducted tests. The DNS servers in BDMS protocol are distributed among different network types. Thus, a lot of propagation delay was produced while translating or resolving addresses among these servers. However, in E4Deliver6 protocol, the border router maintains a mapping table to record the addressing information locally, and this table is always accessed without resolving the alternative DNS server.

It can be noted from Fig. 4(A) and Fig. 4(D) that increasing the number of transmitting nodes in the network with small packet sizes will lead to negatively effect on the performance of these protocols. For example, the variance between 6rd and BDMS in Fig. 4(A) is 1.00827 (ms) and in Fig. 4 (D) is 15.274 (ms) when the packet size =64 bytes. Moreover, the variance in Fig. 4(A) is 0.4888 (ms) and in Fig. 4(D) is 4.6987 (ms) when the packet size = 1024 bytes. Therefore, it is recommended to use large packet sizes (i.e. > 512 bytes) when using one of these protocols.

B. Throughput

Throughput is the amount of data traffic that is transmitted over communication path per unit of time. The simulator measured the Throughput of BDMS, 6rd, and E4Deliver6 protocols with different number of nodes (i.e. 100, 300, 500, 1000 nodes) and packet sizes (i.e. 64, 128, 256, 512, 1024 bytes). In each simulation scenario, the Throughput was measured as the function of the number of transmitting nodes in the local IPv4-only access network at a specific packet size. In all simulation scenarios, the total time that used to simulate each scenario is 100 seconds. The whole network Throughput ($THROUGHPUT_{NETWORK}$) was calculated as the average of all Throughput values for each node individually ($THROUGHPUT_{NODE}$) at a specific packet size. It was calculated as given in the equations Eq. (3) and Eq. (4).

$$THROUGHPUT_{NODE} = \frac{\sum_{i=1}^{50} \frac{N_{Rev(i)} \times P_{size(i)} \times 8}{(T_{end(i)} - T_{start(i)}) \times 1 \times 10^6}}{50} \quad (3)$$

$$THROUGHPUT_{NETWORK} = \frac{\sum_{j=1}^k THROUGHPUT_{NODE}(J)}{K} \quad (4)$$

Where,

$THROUGHPUT_{NODE}$: Throughput value for each node.

$THROUGHPUT_{NETWORK}$: Throughput value for the whole network at specific packet size.

N_{Rev} : number of packets received at destination point.

P_{size} : packet size in bytes.

T_{end} : time at destination host (second).

T_{start} : time at source host (second).

K : total number of transmitting nodes in the network.

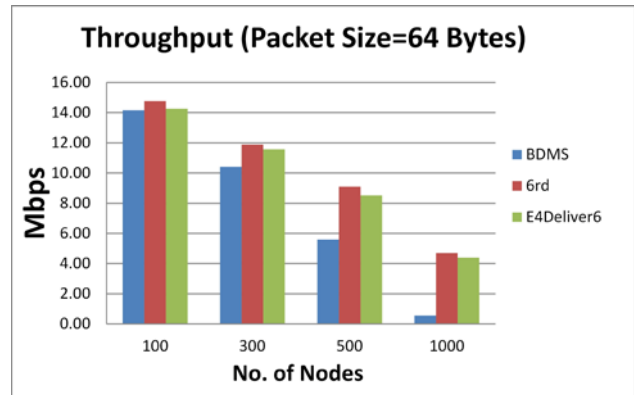


Fig. 5(A) Throughput analysis – 64 bytes

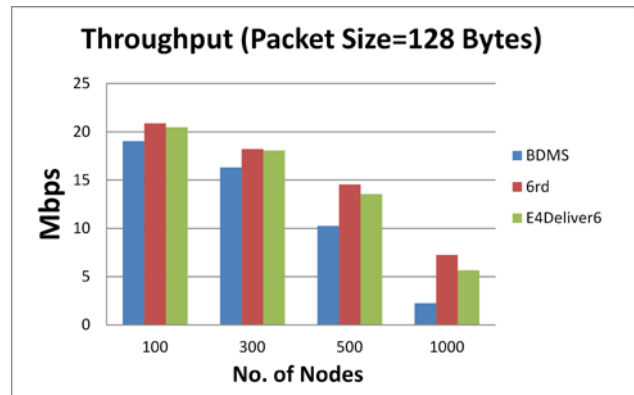


Fig. 5(B) Throughput analysis – 128 bytes

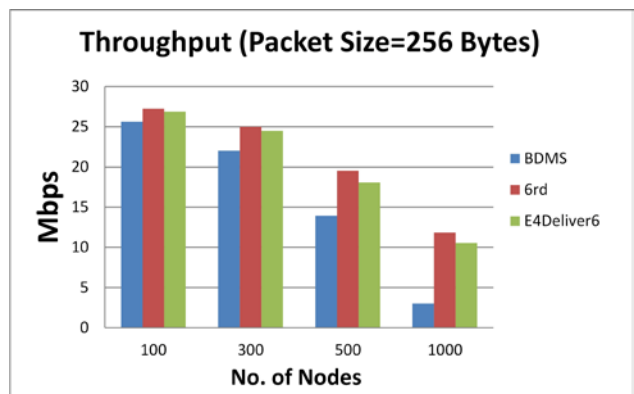


Fig. 5(C) Throughput analysis – 256 bytes

For the sake of accuracy, the $\text{THROUGHPUT}_{\text{NODE}}$ was calculated 50 times for each node. The reported value of $\text{THROUGHPUT}_{\text{NODE}}$ is the average of all recorded values at a specified packet size.

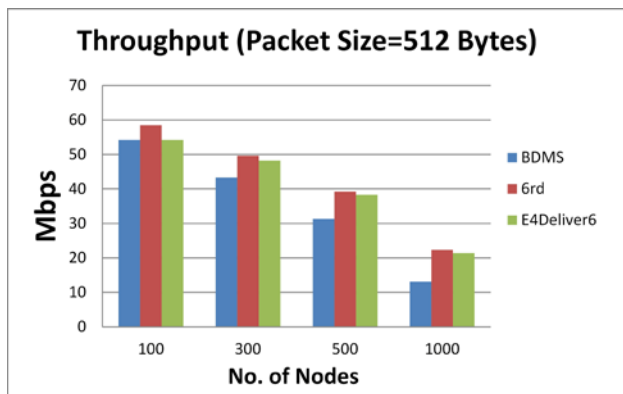


Fig. 5(D) Throughput analysis – 512 bytes

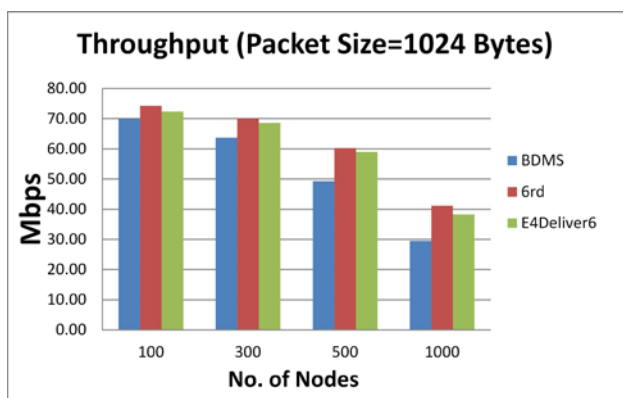


Fig. 5(E) Throughput analysis – 1024 bytes

Fig. 5(A, B, C, D, E) shows a comparative Throughput values for all protocols as the packet size varied from 64 to 1024 bytes.

Traditionally, increasing the number of transmitting nodes in the local access network leads to increase the probability of collision, contention, and delay. Thus, all these factors are negatively effect on the overall Throughput. For example, Fig. 5(A) shows the Throughput values when packet size = 64 bytes. It shows also the significant decrease in Throughput values of BDMS protocol in comparison to other protocols. Increasing number of transmitting nodes in the local access network leads to increase the Throughput gap between BDMS and other protocols especially when transmitting small packet sizes. As mentioned earlier, BDMS is a stateful operation, and it uses different DNS servers to translate and resolve addresses. Additionally, it translates the IP headers of all arrived packets. All these things make BDMS protocol achieved the lowest Throughput values among other protocols. However, as shown in Fig. 5(E), increasing the transmitted packet sizes in BDMS will alleviate the amount of overhead that generated by BDMS operations and hence improve the overall performance. The percentage difference in Throughput values between BDMS and 6rd is decreased from

92.2% in Fig. 5(A) to 28.15% in Fig. 5(E) when number of nodes = 1000. As mentioned earlier, unlike native IPv4-only and IPv6-only based networks, increasing the transmitting packet sizes in these protocols leads to decrease the amount of overhead that is generated in encapsulating, decapsulating, or accessing the mapping tables which reflects a good performance in these networks.

In all conducted tests, the Throughput values of both 6rd and E4Deliver6 protocols are nearly equal. Because 6rd is a stateless operation, it only consumes a small time in encapsulating and decapsulating the received and outgoing traffic. This makes 6rd protocol achieves the best performance among other protocols. Although E4Deliver6 is a stateful operation, it shows acceptable performance parameters in comparison to stateless operation protocols. We clarify this behavior due to the architecture of E4Deliver6 protocols that allows the protocol to maintain a local mapping table to record addressing information of the active connections.

VI. CONCLUSION

This paper introduced a new protocol for ISPs called E4Deliver6. It allows ISPs to continue providing IPv4 service to their customers alongside IPv6 service via E4Deliver6. The E4Deliver6 based hosts will be able to communicate with other IPv6 hosts while they are connected to local IPv4-only access networks. Many simulation tests were conducted in measuring and comparing the performance of E4Deliver6 based network with 6rd and BDMS based networks in terms of latency and throughput parameters. In general, all tests showed acceptable E4Deliver6 performance in comparison to 6rd based networks and much better than BDMS performance parameters. However, the E4Deliver6 protocol showed a small lower performance compared with 6rd protocol due to its stateless operation. Based on the conducted tests, we can conclude that using large packet sizes (i.e. above 512 bytes) in any of 6rd, BDMS, and E4Deliver6 based networks will lead to improve the performance parameters in these networks.

The main drawbacks of this protocol are bottleneck and single point of failure at the TS module. However, the future work might go towards installing multiple TSs in the network. Then, develop an appropriate load balancing algorithm to distribute the IPv6-in-IPv4 tunnels among these TSs. Therefore, this will help in achieving a better quality of service in these networks and avoiding the single point of failure that caused by considering single TS at the border network.

ACKNOWLEDGMENT

The authors would like to thank Prof. Ahmad Al-Qerem the head of the Internet Technology Department at Zarqa University— Zarqa, Jordan for his time dedication, guidance and support during the development of OMNET++ simulator.

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

REFERENCES

- [1] Postel J., "Internet Protocol", Internet Engineering Task Force RFC 791, 1981.
- [2] Deering, S., R., Hinden, "IP Version 6 Addressing Architecture", Internet Engineering Task Force RFC 4291, 2006.
- [3] S. Kent, K. Seo, "Security Architecture for the Internet Protocol", Internet Engineering Task Force RFC 4301, 2005.
- [4] D. Žagar, G. Martinović and S. Rimac-Drlje, "Testing the Tools for IPv6 Traffic Tunneling", Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, 2005.
- [5] M. Badamchizadeh and A. Chianeh, "Security in IPv6", Proceedings of the 5th WSEAS International Conference on Signal Processing, Istanbul, Turkey, May 27-29, 2006.
- [6] A. Hamarshah, M. Goossens, R. Alasem, "Decoupling Application IPv4/IPv6 Operation from the Underlying IPv4/IPv6 Communication (DAC)", American Journal of Scientific Research, Eurojournals Press, Issue 14, 2011, PP. 101-121.
- [7] J. Chen, Y. Chang, C. Lin, "Performance Investigation of IPv4/IPv6 Transition Mechanisms", Journal of Internet Technology, Volume 5 No.2, 2004, pp. 163-170.
- [8] B. Forouzan, "TCP/IP Protocol Suite"; McGraw-Hill, 2003.
- [9] J. F. Kurose, K. W. Ross, "Computer Networking A Top-Down Approach Feaching the Internet", Pearson Education, New York, 2003.
- [10] N. Oliver, V. Oliver; "Computer Networks Principle, Technologies and Protocols for Network Design", John Wiley and Sons, England, 2006.
- [11] J. Chen, Y. Chang, C. Lin, "Performance Investigation of IPv4/IPv6 Transition Mechanisms", Journal of Internet Technology, Volume 5 No.2, 2004, pp. 163-170.
- [12] D. Zagar, G. M. and S. Rimac-Drlje, "Security Analyses of IPv4/IPv6 Tunneling Tools", WSEAS TRANSACTIONS on COMPUTERS , Issue 1, Volume 5, 2006.
- [13] K. OH, K. LEE, Y. WON, "Malicious Code Detection Method over IPv4/IPv6 Tunneling Using Naive Bayesian Classifier", Proceedings of the 5th WSEAS International Conference on Information Security and Privacy, Venice, Italy, 2006.
- [14] W. Townsley, O. Troan, " IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) -- Protocol Specification", Internet Engineering Task Force RFC 5969, 2010.
- [15] R. Despres, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)", Internet Engineering Task Force RFC 5569, 2010.
- [16] R. AlJa'afreh, J. Mellor, M. Kamala, B. Kasasbeh, "Bi-Directional Mapping System as a New IPv4/IPv6 Translation Mechanism", Tenth International Conference on Computer Modeling and Simulation, 2008.
- [17] S. Deering, "ICMP Router Discovery Messages", Internet Engineering Task Force RFC 1256, 1991.
- [18] F. Baker, X. Li, C. Bao, K. Yin, "Framework for IPv4/IPv6 Translation", Internet Engineering Task Force Draft, 2010.
- [19] Bao C., Huitema C., Bagnulo M., Boucadair M., Li X., "IPv6 Addressing of IPv4/IPv6 Translators", Internet Engineering Task Force RFC 6052, 2010.
- [20] E. Nordmark, R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", Internet Engineering Task Force RFC 4213, 2005.
- [21] Internet Architecture Board, "IAB Technical Comment on the Unique DNS Root", Internet Engineering Task Force RFC 2826, 2000.
- [22] M. Bagnulo, P. Matthews, I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", Internet Engineering Task Force Draft, 2010.
- [23] "OMNeT++ Network Simulation Framework", Available: <http://www.omnetpp.org>.
- [24] I. Raicu, S. Zeadally, "Evalating IPv4 to IPv6 transition mechanisms". IEEE International Conference on telecommunications, 2003.
- [25] Isabel Dietrich, "OMNET++ Traffic Generator", Available: <http://www7.informatik.uni-erlangen.de/~isabel/omnet/modules/TrafGen/>

Ala Hamarshah is a PhD candidate at the faculty of Engineering, Vrije Universiteit Brussel – Belgium. He has been awarded a full scholarship from Erasmus Mundus ECW II project. Prior this, Hamarshah was working as a full time instructor at the Arab American University - Palestine. He received his BSc of Computer Science at the faculty of Science, Birzeit University – Palestine 2000. He gained his MSc in Computer Science at the Kind Abdullah II School for IT, The University of Jordan 2003 – Jordan. His research focuses on proposing techniques and mechanisms to break the deadlocks for transitions to IPv6.

Marnix Goossens graduated in electrical and mechanical engineering (Masters) in 1978, and obtained a PhD in engineering in 1985, both at the Vrije Universiteit Brussel, Belgium. He became full-time Professor – teaching and research - at the same University afterwards, and heads since a research group working on digital telecommunications, especially "protocol engineering". The research team has much collaboration with industry, and has a tradition in research on practical oriented communication problems.