Configuring Hosts to Auto-detect (IPv6, IPv6-in-IPv4, or IPv4) Network Connectivity

Ala Hamarsheh[1*], Marnix Goossens[1] and Rafe Alasem[2]
[1]Department of Electronics and Informatics ETRO, Vrije Universiteit Brussel
Office: 4k222, Building K, Pleinlaan 2, 1050 Elsene, Belgium
[e-mail: {ala.hamarsheh, marnix.goossens}@vub.ac.be]
[2]Computer Science Department, Imam Muhammad ibn Saud Islamic University
Office No: FR-90, Riyadh, 11432, Saudi Arabia
[e-mail: rafe.alasem@gmail.com]
* Corresponding author: Ala Hamarsheh

Received April 6, 2011; revised June 7, 2011; accepted June 24, 2011; published July 28, 2011

Abstract

This document specifies a new IPv6 deployment protocol called CHANC, which stands for Configuring Hosts to Auto-detect (IPv6, IPv6-in-IPv4, or IPv4) Network Connectivity. The main part is an application level tunneling protocol that allows Internet Service Providers (ISPs) to rapidly start deploying IPv6 service to their subscribers whom connected to the Internet via IPv4-only access networks. It carries IPv6 packets over HTTP protocol to be transmitted across IPv4-only network infrastructure. The key aspects of this protocol are: offers IPv6 connectivity via IPv4-only access networks, stateless operation, economical solution, assures most firewall traversal, and requires simple installation and automatic configuration at customers' hosts. All data packets and routing information of the IPv6 protocol will be carried over the IPv4 network infrastructure. A simple application and a pseudo network driver must be installed at the end-user's hosts to make them able to work with this protocol. Such hosts will be able to auto-detect the ISP available connectivity in the following precedence: native IPv6, IPv6-in-IPv4, or no IPv6 connectivity. Because the protocol does not require changing or upgrading customer edges, a minimal cost in the deployment to IPv6 service should be expected. The simulation analysis showed that the performance of CHANC is pretty near to those of native IPv6, 6rd, and IPv4 protocols. Also, the performance of CHANC is much better than that of D6across4 protocol.

Keywords: IPv6 deployment, IPv4-only access networks, Firewall traversal, Application level tunneling

DOI: 10.3837/tiis.2011.07.002

1. Introduction

The Internet protocol (IP) that is mainly used in the Internet and networks is IP Version 4 (IPv4 [1]). This protocol was a tremendous success for over 30 years and has played a great role in the Internet revolution. However, regardless of all these credits, IPv4 developers did not envision nowadays communication's requirements such as scalability (address space capacity), flexibility, Quality of Service (QoS), security (IPsec [2]), etc.

As for limitations on scalability, the Internet Engineering Task Force (IETF) has proposed a set of mechanisms (i.e. NAT [3] and CIDR [4]) to alleviate the scarcity of public IPv4 addresses. For example, NATs are deployed to translate the public IPv4 into a set of private IPv4 addresses; hence it allows reusing the IPv4 private address space.

All these solutions are makeshift measures to extend the life of IPv4 address space. They will not ultimately overcome the scarcity of IPv4 public address space and many other today communication's limitations. The intensive use of network based devices, alongside with the growth of the Internet and networking technologies, lead to exhaust the IPv4 public address space. These reasons drove the Internet society to develop a new addressing scheme (i.e. IPv6 [5]) in order to resolve all these issues. IPv6 comes with 128-bit address space, simple header format, efficient routing, support both stateless and stateful address configurations (i.e. in stateless address auto-configuration, there will be no manual configurations at the IPv6 hosts. Therefore, these hosts will be responsible for creating their own addresses. In stateful address auto-configuration, hosts obtain their addresses and configuration information from a certain server), built-in security, QoS, etc. [6]. Due to the size of the Internet, there will be no flag day for transition from IPv4 to IPv6. IPv6 protocol will slowly and gradually spread into networks and the whole Internet. Therefore, IPv4 and IPv6 will coexist for a certain amount of time. Unfortunately both IPv4 and IPv6 protocols are incompatible because the addressing system and datagram format are different. The IETF has created a set of working groups to smooth IPv6 transition and it has proposed a lot of pragmatic solutions to achieve it. These solutions can be categorized into dual stack network (hosts and routers), tunneling, and protocol translation [7][8][9][10].

End-users are not concerned with the transition itself, but unfortunately there are implications in the transition for those end-users. The authors have proposed a mechanism called DAC [11] to overcome one of these implications, being that applications also need to be transformed for IPv6 due to the different address arise. DAC allows IPv4-only applications to work over IPv6.

The problem, which this paper attempts to address, is how to deploy IPv6 service across IPv4-only access networks rapidly, automatically, effectively, and with minimum cost possible. The fewer and fewer available IPv4 addresses and the big depletion in IPv4 address space make ISPs face serious challenges to continue providing IPv4 service to their customers. In America, Europe, and Africa, the Regional Internet Registries (RIR) allocated additional IPv4 addresses from the central pool. On February 3, 2011 IANA has allocated the last IPv4 addresses to RIR [12]. Experts predict that the RIRs will be out of address later in 2011. The depletion of IPv4 address space will force many ISPs to implement and deploy their IPv6 network architecture more quickly than they would like. This may cause a disastrous and long term consequences.

Providing IPv6 to subscribers require changing/upgrading IPv4-only access network infrastructure, as well as doing some manual configurations at customers' hosts. However, the proposed protocol assures IPv6 connectivity without changing or upgrading current IPv4-only network infrastructure (i.e. minimal cost expected). Also it assures automatic configurations at customers' hosts. CHANC intends to make the transition to the parties involved (end-users and network providers) runs concurrently and transparently. It configures hosts to make them able to auto-detect all available network connectivity types provided by their ISPs, automatically locates the desired relay server, tunnels the outgoing IPv6 traffic into HTTP-in-IPv4 packets, and then transmits them over IPv4 network infrastructure. The mechanism also intends to make the ISPs be able to provide different network connectivity types (native IPv6 or tunneled IPv6 and IPv4) for individual hosts.

The most important aspect of this protocol is its ability to provide the IPv6 connectivity to individual hosts rather than sites. Depending on the host configurations, customer site could have multiple hosts; these hosts may communicate over distinct connectivity types.

The advantages of CHANC protocol are:

- ISPs can easily start deploying IPv6 connectivity while they do not or cannot support native IPv6 connectivity to their subscribers due to IPv4-only access network.
- Stateless Operation: The stateless based translation mechanisms are used to support end-to-end address transparency when connections are initiated from IPv4 to IPv6 networks and vise-versa. These mechanisms use algorithmic mapping between IPv4 and IPv6 addresses (i.e. IPv4-compatible IPv6 addresses [6]) to translate IPv4 addresses into IPv6 and vise-versa. However, stateful based translation mechanisms maintain a translation state to map IPv4/port number pairs with IPv6/port number pairs [13].
- Economical solution, it does not require changing/upgrading IPv4-only access network infrastructure. CHANC does not try to deal with the issues of microeconomics theories such as [14]. Instead, CHANC tries to reduce the cost from the viewpoint of IPv4 access network infrastructure (i.e. it does not require changing/upgrading the local IPv4-only access network for both the ISP and even more important, the end-user).
- Automatic configurations at the subscribers' hosts.
- CHANC traffic looks like normal IPv4 traffic. Hence, the proposed protocol will not modify or change anything in the network layer.
- CHANC generated traffic looks like web traffic (i.e. because it is encapsulated in HTTP protocol), this allows the traffic to traverse most firewalls that barring tunneled packets (packets with protocol ID = 41).
- Each ISP can delegate its own IPv6 prefixes(s). This allows them to closely control and manage their IPv6 traffic which leads to better network management and quality of service.
- CHANC may be discontinued, and the connectivity can be moved to native IPv6 once the the network and customers' hosts can fully support native IPv6 access and transport.

It is important to highlight here that CHANC based hosts adopt stateful address auto-configurations to obtain their IPv6 addresses. But, CHANC protocol is a stateless operation in translating addresses between IPv4 and IPv6 networks.

Fig. 1 demonstrates the protocol infrastructure while being deployed by a certain ISP. The proposed protocol allows hosts to auto-detect the connectivity in the following precedence: IPv6, IPv6-in-IPv4, or native only IPv4 connectivity. The host can then automatically configure itself (see later) to communicate over one of the available connectivity types by following the previous precedence.

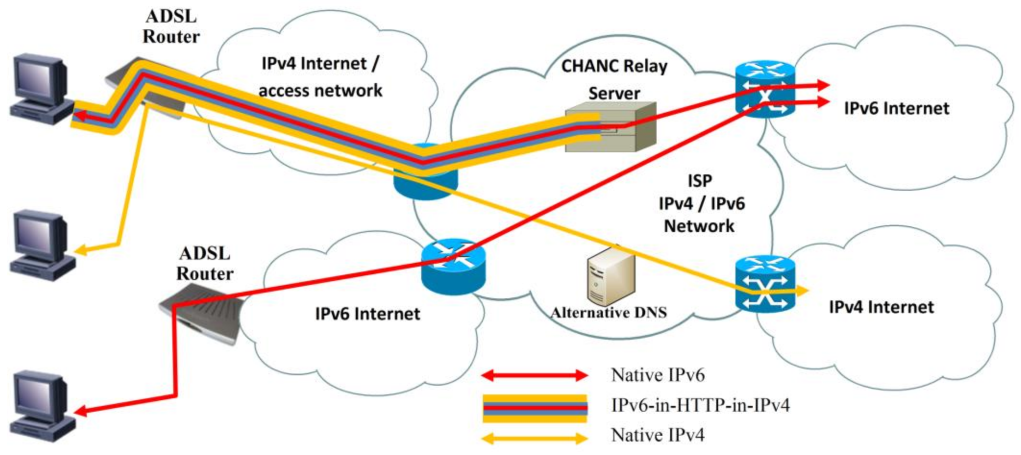

Fig. 1. CHANC infrastructure

2. Related Work

2.1 IPv6 Rapid Deployment (6rd)

6rd protocol [15][16] intends to deploy IPv6 service across local IPv4 access networks by tunneling IPv6 traffic into IPv4 traffic and transmitting it over IPv4 network infrastructure. 6rd uses Network Specific Prefixes (NSP), then each ISP can deploy its own prefixes. This allows them to closely manage their IPv6 traffic. It allows tunnels to be created between service provider's border relay (BR) and customer edge (CE). 6rd is a stateless address auto-configuration protocol. Thus, 6rd hosts will be responsible for creating their IPv6 addresses. The 6rd delegated IPv6 prefix is calculated by CE during 6rd configurations and sub-delegates this prefix to the customer's devices. The followings are the protocol's requirements:

1. Changing/upgrading all CEs, knowing that the CE must be configured with the following elements:
    a. 6rd prefix (i.e. IPv6 prefix for a certain 6rd domain)
    b. 6rdPrefLen (i.e. the length of 6rd prefix)
    c. IPv4MaskLen (i.e. the most significant bits that are identical across all IPv4 CE address within 6rd domain)
    d. 6rdBRIPv4Address (i.e. the IPv4 address of the border router)
2. Installing border relay router at the ISP side.
3. Configuring a new DHCPv4 OPTION-6RD (212) to deliver all 6rd elements to the CEs.

The main limitations of 6rd protocol are:

1. It requires changing/upgrading CEs which will increase the cost of IPv6 deployment at both ISPs and end-users sides.
2. The protocol does not address the situation if there are more than one level of DHCPv4 servers between BR and CE.
3. 6rd must be configured at CE, thus changing the IPv6 transition plan to something else would require changing/configuring the CEs again.

4. The tunneled traffic cannot traverse firewalls (i.e. most firewalls block traffic with protocol number 41).
5. CEs that support 6rd is still lacking.

2.2 Deploying IPv6 Service across Local IPv4 Access Networks (D6across4) (by the authors of this paper)

D6across4 [17] is a stateless address auto-configuration, as well as a stateful operation protocol.

The idea behind this protocol is to configure hosts rather than customer edges. Thus, tunnels can be created between ISP's tunnel server and customers' hosts. Since it does not use DHCPv4 to deliver the address configuration elements, D6across4 can work in networks where multilevel of DHCPv4 servers exists. IPv6 addresses that assigned to D6across4 hosts are virtual IPv6 addresses (i.e. un-routable addresses). The tunnel servers maintain a state to track the communication sessions for all active connections. Tunnel servers work like a NAT in terms of replacing IPv6 virtual addresses of outgoing traffic by tunnel server's IPv6 address.

The network administrator must configure the local DNS by tunnel server's IPv4 address. Therefore, tunnel endpoints can be determined automatically by resolving tunnel servers' domain names.

AINA must assign a 28-bit well-known prefix to be used in creating D6across4 virtual and tunnel servers' IPv6 addresses as well.

The main limitations of D6across4 protocol are:

1. D6across4 is a stateful operation; therefore, tunnel servers consume extra overhead in accessing mapping tables at every packet received.
2. It can be used in small networks that consist of few nodes. However, deploying it at large scale can lead to scalability issues that could heavily limit the IPv6 performance compared to native solutions.
3. As it uses well-known IPv6 prefixes, this will reduce the chance to allow ISPs manage and control their traffic.
4. Similarly to 6rd protocol, D6actoss4 traffic cannot traverse firewalls.

The common key aspect behind all these protocols (i.e. 6rd, D6across4, and CHANC) is to allow for access to IPv6 service when ISPs' access networks do not or cannot provide native IPv6 connectivity. **Table 1** compares between these three protocols.

Table 1: comparison between 6rd, D6across4, and CHANC protocols

	No Host Config.	Stateless	No CE Config.	Appl. Level Tunnelin g	Flexible IPv6 Prefixing	Traverse Firewalls	Economi-cal Solution	Good Perform-ance
6rd	✓	✓			✓			✓
D6acros s4			✓				✓	
CHANC		✓	✓	✓	✓	✓	✓	✓

3. Protocol Specifications

The CHANC protocol aims to deploy IPv6 service relaying over the IPv4-only access networks. The IPv4-only access networks considered as transport layer to deliver IPv6-in-HTTP traffic to CHANC based hosts. Unlike 6rd and D6across4 protocols, CHANC uses application level tunneling which allows more simplicity, abstraction, firewall travseral, and control in the forwarded IPv6 traffic. For example, CHANC uses HTTP protocol as a carrier protocol (explained in section 3.1.1). Therefore, we will be guaranteed that the traffic will traverse most simple firewalls. Moreover, using such type of tunneling will hide all the complexities in the network layer (i.e. MTU, fragmentations, etc.). The protocol establishes automatic tunnels between hosts and CHANC Relay Server (CRS) device to carry IPv6-in-HTTP-in-IPv4 traffic. It relies upon algorithmic mapping between IPv4 and IPv6 addresses allocated at the ISP side and assigned to CHANC based hosts. Tunnel endpoints are automatically determined by resolving DNS for CRS IPv4 address.

CHANC based network consists of CRS, CHANC based hosts, and alternative DNS [18]. The CHANC based host is able to send/receive IPv6 traffic. When it starts up, firstly it should obtain IPv6 address, and then try to resolve the CRS server's IPv4 address to post the IPv6 traffic via HTTP to this server. Because the host is connected via IPv4-only access network, encapsulation and decapsulation of DHCPv6 [19] messages should be applied as well while communicating with DHCPv6 server. Thereafter, all outgoing IPv6 packets will be encapsulated within HTTP-in-IPv4 and posted to the IPv4 network. CRS, on the other hand, upon receiving these encapsulated packets, it will extract the IPv6 packets and then inject them unmodified into IPv6 network interface. However, not all received IPv6 packets will be injected into IPv6 network. For example, the DHCPv6 messages will be forwarded to internal the DHCPv6 server. The CRS will forward the received IPv6 traffic to the CHANC based hosts unmodified throughout HTTP protocol as web traffic. The CRS works like a router in terms of forwarding the received traffic, but with the additional functionality of extracting the received tunneled packets from CHANC based hosts and forwarding these IPv6 packets to the Internet. Similarly, the CRS will receive IPv6 packets from the Internet and post them to the CHANC based host(s) (see further).

The protocol may be discontinued once the ISP recognizes that the network and all customers' hosts can fully support native IPv6 access and transport.

Deploying this protocol requires some configurations at ISP side and end-users hosts. CHANC based hosts will automatically try to configure their network connectivity according to the following precedence:

1. Try to communicate using native IPv6 connectivity.
2. If the first option fails, then try IPv6-in-HTTP-in-IPv4 connectivity.
3. If all the previous trials fail, then try to communicate over native IPv4 connectivity.

Some network components must be installed and some configuration must be done at the ISP side prior deploying this protocol. The local host should be able to detect the existence of CHANC service at the ISP. After that, it will be possible for that host to auto-configure its network drivers to allow IPv6 traffic to be transmitted over the available network connectivity by following the previous precedence.

3.1 CHANC Based Host Architecture

As mentioned earlier, CHANC Based Hosts (CBH) appear as dual stack hosts outside their IPv4-only access networks. Besides, all IPv6 outgoing traffic that destined to CRS will be transmitted as HTTP-in-IPv4 traffic. When the tunneling client detects the existence of CHANC protocol, it starts posting IPv6 traffic to CRS server as web traffic. The IPv4 applications can communicate through IPv4 stack, and the IPv6 applications can communicate through IPv6 stack. As for IPv6 applications while trying to communicate through IPv6 stack, the pseudo network driver will sniff all the IPv6 packets and forward them to tunneling client application. The tunneling client then will post the unmodified IPv6 traffic to CRS throughout HTTP protocol as web traffic. The host architecture consists of the following modules: Tunneling client and Pseudo network driver. Fig. 2 illustrates CBH architecture.

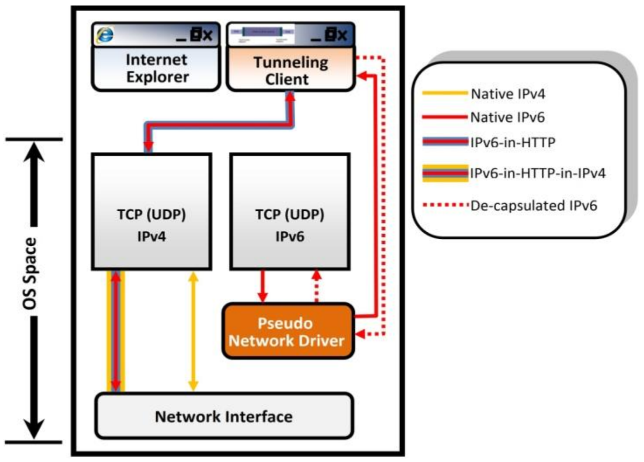

Fig. 2. CBH architecture

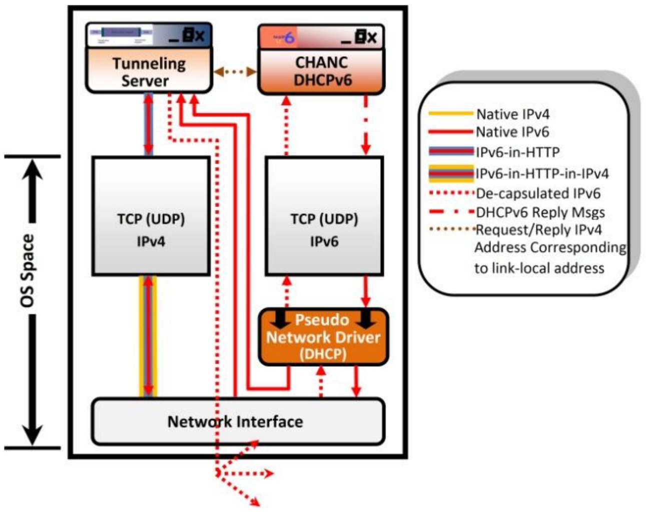

Fig. 3. CRS architecture

3.1.1 Tunneling Client:

It is an application running at the CBH side. It has the ability to receive IPv6-in-HTTP and IPv6 packets directly from CRS and pseudo network driver respectively, then either it extracts the IPv6 packets from IPv6-in-HTTP traffic and injects them into pseudo network driver, or post the received IPv6 packets to CRS as web traffic.

The presence of firewalls between CBH and CRS poses a potential problem. This problem is barring tunnels to be established throughout firewalls. This is caused by the fact that a firewall typically blocks all IP traffic with protocol ID = 41. To overcome this problem, the HTTP/1.1 [20] is used in this protocol as a carrier protocol. HTTP is used because it is probably the only protocol for which we can be sure that it will be granted for most firewall traversal. A simple request/response can be implemented on the top of HTTP via XML [21] page. The HTTP POST message with format XML encoded parameters used to transport contents (i.e. raw IPv6 packets), and CBH IPv6 link-local address, CBH IPv4 address, and CE IPv4 addresses from CBH to CRS. These addresses will be extracted and used at the CRS side to create CBH's IPv6 address.

The basic functionality is that, initially the CBH will try to communicate with the CRS via TCP/IPv6, if this fails, the HTTP will be used to post IPv6 packets. In order not to modify the architecture too much we will carry the unmodified IPv6 packets inside HTTP. In all HTTP communications, the HTTP POST will be used because this defines a way to carry arbitrary blocks of data (IPv6 traffic) transmitted from CBH to CRS and vice-versa.

The tunneling client does the following actions:

1. Checking if the current ISP is providing CHANC protocol or not: all tunneling clients are configured with the same CRS domain name. The tunneling client will try to resolve this domain name. Receiving a reply (i.e. CRS's IPv4 address) is an indication of the availability of the service.
2. Receiving IPv6 packets from pseudo network driver and posting them to CRS server via HTTP protocol like any web traffic destined to web server. Some HTTP header fields must be set with arbitrary values to make the communication with HTTP proxies handled better.
   - Date Header: set a value at an arbitrary point in the past to keep proxies from caching.
   - Expires Header: the same date header value.
   - Pragma: No-Cache.
   - Cache-Control: No-Cache.
   - Content-type: application/CHANC-tunneling-client
3. Encoding the addresses in the XML page as follows:
   ```
   <CBH_IPv4> CBH IPv4 address </CBH_IPv4>
   <CBH_LLA> CBH IPv6 Link-local address </CBH_LLA>
   <CE_IPv4> CE IPv4 address </CE_IPv4>
   ```
4. Posting IPv6-in-HTTP traffic to the CRS via IPv4 communication stack.
5. Extracting the IPv6 payloads from received IPv6-in-HTTP data, and directly injecting these unmodified IPv6 packets into pseudo network driver (they will appear as they received via IPv6 stack).

This module receives all types of IPv6 traffic from pseudo network driver including IPv6 payload, ICMPv6, DHCPv6, etc., then post them to CRS via HTTP protocol unmodified. However, the Maximum Transmission Unit (MTU) and fragmentation issues will be handled by TCP and they are away of scope of this document.

3.1.2 Pseudo Network Driver:

It has the ability to sniff all IPv6 packets received via TCP/IPv6 stack and forward them to the tunneling client application. Moreover, it receives the unmodified IPv6 packets delivered from tunneling client and injects them into TCP/IPv6 stack. It is important to note here that all IPv6 traffic will be forwarded to the tunneling application including control information (DHCPv6, ICMPv6, etc.). The CBH IPv6 based applications can call IPv6 socket APIs functions normally and communicate throughout IPv6 stack as they natively connected to IPv6 network. The following are the algorithms of tunneling client and pseudo network driver modules:

Algorithms: Tunneling Client & Pseudo Network Driver

```
                    ** Tunneling Client **

//test if host is connected to IPv6 network
CRS_address ← getnameinfo("Resolve Any IPv6 Host")
If CRS_address ≠ nil Then
  // the host is connected to IPv6 network, pass all IPv6
  // unmodified
ElseIf (CRS_address ← gethostbyname("CRS-Address"))≠ nil Then
  // test if host is connected to CHANC network
  v6_pkts[] ← nil
     While (( v6_pkts[] ← receive()) ≠ nil)
         page_para ← nil
         Select Case v6_pkts[0].source_address
                Case CBS.source_address:
                     setHeader ("date", "1-1-1990")
                     setHeader ("expires", "1-1-1990")
                     setHeader ("pragma", "no-cache")
                     setHeader ("cache-control ", "no-cache")
                     setHeader ("content-type", "application
                                  /CHANC-tunneling-client")
                     Page_para = "<CBH_IPv4>" + CBH_IPv4 +
                                 "</CBH_IPv4>" + "<CBH_LLA>" +
                                  CBH_LLA + "/<CBH_LLA>" +
                                  "<CE_IPv4>" + CE_IPv4 +
                                  "</CE_IPv4>"
                     Send_over_HTTP(v6_pkts,page_para,
                                               CRS_address)
                Case Else
                     Pseudo_network_driver.inject(v6_pkts)
         End Select
     Wend
Else
 // all previous tests failed. The host is connected
 // to IPv4 network and the ISP does not support CHANC.
 // initialize time to perform all previous tests periodically
End If
                       ** Pseudo Network Driver **

while (( v6_pkts[] ← receive()) ≠ nil)
   If (v6_pkts[0].source_address ≠ CBH.source_address) Then
       TCPv6.inject(v6_pkts)
   Else
       Tunnel_client.forward(v6_pkts)
   End If
Wend
```

3.2 ISP Configurations

The ISPs can continue providing IPv4 connectivity to their customers alongside IPv6 connectivity using CHANC protocol. Some users will be upgraded to work over CHANC and others will not. Hence, the ISPs should be able to provide different connectivity types (IPv4 and IPv6). The customers can gradually upgrade their hosts to communicate over CHANC protocol. This will lead to ease and smooth the process of transition to IPv6 connectivity. The ISPs should install and configure their access network with the following components: CHANC Relay Server (CRS) and Alternative DNS.

3.2.1 CHANC Relay Server:

The CHANC Relay Server (CRS) works like a router in terms of packets forwarding. It has the ability to receive IPv6-in-HTTP-in-IPv4 traffic, extract IPv6 packets from received traffic, and forward them to IPv6 Internet. Moreover, it has the ability to receive the IPv6 packets from IPv6 Internet, and forward them to CBH(s) as IPv6-in-HTTP-IPv4 traffic. The CRS is configured to allocate IPv6 addresses to CBHs when they startup. The CRS is a multifunctional device which consists of: Tunneling Server, CHANC DHCPv6, and Pseudo Network Driver (DHCP). **Fig. 3** illustrates CRS architecture.

- **Tunneling Server**

The Tunnel server is an application running at the CRS side. It is capable of receiving IPv6-in-HTTP payload, as well as sniffing IPv6 packets arrived from CBH and IPv6 Internet respectively. The tunneling server extracts the IPv6 packets from the received IPv6-in-HTTP payloads, and then injects them directly into network interface. It maintains a temporary mapping table with lease time field, this table maps CBH IPv4 and CE IPv4 with CBH link-local address after extracting them from HTTP header (see section 4.1.1). This information will be requested by DHCPv6 server at the time of creating IPv6 addresses. Because the tunnel server maps these addresses for every incoming request, the network administrator should carefully set the lease time value in the mapping table with a smallest possible value (i.e. it can be equal to the maximum time that IPv6 host takes to obtain its IPv6 address from DHCPv6 server).

The extracted IPv6 packets must appear as originated from CBH host outside ISP IPv4-only access network. In order to keep the source addresses of these IPv6 packets unchanged, the tunneling server must not forward these packets via CRS TCP/IPv6 stack. Instead, these packets will be injected directly into CRS's network interface. The tunneling server also is configured to respond to CHANC DHCPv6 server when request the tunneling server for IPv4 addresses corresponding to a certain link-local address, these addresses will be used to create IPv6 addresses before assigning them to the CBHs.

The tunneling server application sniffs all incoming IPv6 packets directly from network interface. Then it tries to extract the IPv4 address from the destination IPv6 address. If IPv4 address cannot be extracted (i.e. destination address is a link-local address), then it looks up in the mapping table trying to retrieve the corresponding IPv4 address in terms of the received link-local address. Finally, it posts these IPv6 packets to CBH via HTTP protocol using extracted or retrieved IPv4 address. The outgoing traffic will be like any web traffic destined to CBH. We notice here that the size of mapping table will be very small because of short lease time field value of the mapping table. Additionally, this table is referenced only while assigning IPv6 addresses to CBHs only. Therefore, accessing this table will not effect on the performance of CHANC.

- **CHANC DHCPv6:**

The CBHs are connected to IPv4-only access network. Thus obtaining IPv6 address through DHCPv6 is tricky and not a traditional way. CRS is configured with internal DHCPv6 server to allocate IPv6 addresses to CBHs. Usually CBHs continue exchanging DHCP messages with DHCPv6 server before and even after obtaining their IPv6 addresses (i.e. SOLICIT, ADVERSTISE, REQUEST, RENEW, REBIND, etc.). All these messages will be posted over

HTTP protocol and transmitted to DHCPv6 server over IPv4 infrastructure. DHCPv6 server on the other hand is configured with an IPv6 Network Specific Prefix (NSP) [22]. This prefix is used in creating IPv6 addresses. It is a paramount importance to highlight here that each ISP can delegate its own IPv6 prefixes(s), this makes them closely control and manage their IPv6 traffic and leads to better network management and quality of service. The IPv6 address is created by concatenating NSP prefix with subnet ID (left zeros for future use), CBH IPv4 address, and CBH interface ID. The network administrator must configure DHCPv6 server with that NSP. In order to create IPv6 address, the DHCPv6 server needs CBH's IPv4 address. Thus it queries the tunneling server about CBH IPv4 address in terms of link-local address. Finally, it concatenates NSP with subnet ID, CBH IPv4 address, and interface ID to form the IPv6 address. **Fig. 4** shows the format of IPv6 address created by CHANC DHCPv6 server.

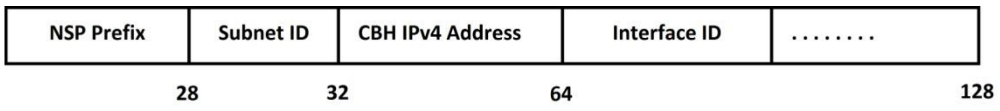

Fig. 4. IPv6 address format

CRS and CHANC DHCPv6 server must use the same NSP and prefix length. This makes all the generated IPv6 addresses have the same CRS IPv6 prefix. Thus, all incoming traffic that destined to CBHs will be routed to CRS server first. The DHCPv6 server must be configured with all these extensions to be CHANC DHCPv6. The CHANC DHCPv6 functions are: (1) allocate synthesized IPv6 addresses by concatenation NSP with CBHs' IPv4 addresses. (2) query tunneling server for IPv4 in terms of link-local address.
The current versions of the DHCPv6 source code can be found here [23].

- **Pseudo Network Driver (DHCP):**

Like client pseudo network driver, this module is capable of sniffing the DHCPv6 packets that are received via TCP/IPv6 stack, and then forwarding them to the tunneling server application. It is important to highlight here that this module sniffs the traffic that is generated by CHANC DHCPv6 server only. Therefore, the module is configured to forward any received IPv6 traffic to the tunneling server application unmodified.

The following are the algorithms of tunneling server and pseudo network driver (DHCP) modules:

Algorithms: Tunneling Server & Pseudo Network Driver (DHCP)

```
                ** Tunneling Server **

//test all kinds of received packets and take the appropriate
//action
pkts[]← nil
While true
  Pkts[]← receive()
  SRC_address ← pkts[0].source_address
  DST_address ← pkts[0].destination address
  If SRC_address = LINK-LOCAL Then  // extract addresses
    CBH_IPv4 ← XML.CBH_IPv4        // from XML page
    CBH_LLA ← XML.CE_LLA
    CE_IPv4 ← XML.X-CE_IPv4             // LT : Lease Time
    Insert into Mtable Values (CBH_IPv4,CBH_LLA,CE_IPv4,LT)
    Pseudo_network_driver.inject(pkts)
  ElseIf SCR_address = CRS_address AND
                     DST_address = LINK-LOCAL Then
```

```
              Select CBH_IPv4,CE_CE From Mtable // retrieve IPv4 address
                    Where CBH_LLA = LINK-LOCAL  // corresponding to LLA
              Send(CBH_IPv4, pkts)
            ElseIf startsWith(SRC_address, NSP) Then // outgoing traffic
              Pseudo_network_driver.inject(pkts)
            Else // incoming traffic from IPv6 Internet
              CBH_IPv4 ← Extract (DST_address, 32, 64)
              Send(CBH_IPv4, pkts)
            End If
          Wend

          void Send (Address CBH_IPv4, Packets pkts){
               setHeader ("date", "1-1-1990")
               setHeader ("expires", "1-1-1990")
               setHeader ("pragma", "no-cache")
               setHeader ("cache-control ", "no-cache")
               setHeader ("content-type", "application
                                       /CHANC-tunneling-client")
               Send_over_HTTP(pkts, CBH_address)
          }

          ** Pseudo Network Driver (DHCP) **

          while ((pkts[] ← receive()) ≠ nil)
                 Tunnel_server.forward(v6_pkts)
          Wend
```

3.2.2 Alternative DNS:

When ISPs provide different type of services (IPv4 and IPv6), they must configure their DNS properly to make it capable to resolve any type of 'A' and 'AAAA' [24] records resolutions. Also it must be configured manually with CRS IPv4 address and domain name to make it resolvable by CBHs. The policy of CHANC protocol states that the CRS domain name must be common between all ISPs that they use CHANC protocol to communicate with other servers located in the IPv6 Internet. In such a way, the network administrator can assign any IP address to denote for the CRS domain name. This is considered a very simple operation as the network administrators can configure their DNS servers only once at the time of setup.

4. Performance Analysis

The purpose of the simulation was to prove the validity and evaluate some of the important performance parameters of the proposed protocol with other related protocols in large scale settings. We reported on the CHANC performance over several simulated scenarios. CHANC was analyzed under synthetic traffic generator. This traffic was generated using TrafGen [25] tool as it has the ability to generate TCP traffic between client/server and to define traffic parameters for a certain traffic flow (i.e. packet size, destination host, generating traffic with ON and OFF in seconds, etc.). Different tests were conducted to measure and compare CHANC based network performance with the performance of 6rd, D6across4, IPv6, and IPv4 based networks. The simulator was implemented on different network types (6rd, D6across4, CHANC, IPv6, and IPv4 based networks) with different number of hosts in each access network.

We used OMNET++ [26] as a simulation platform. OMNET++ is a public source, component-based, modular, object oriented, and open-architecture discrete event network simulator. INET framework [27] is also used to implement IPv4, IPv6, and TCP protocols. The INET framework builds upon OMNET++.

In real implementation of CHANC, CBH will post the IPv6 packets over HTTP using tunneling client application. This will lead to hide all the network complexities and finally improve the overall performance by selecting the appropriate MTU. In our simulation, the performance of CHANC protocol was tested by making CBHs transmitting packets with specified size for a certain amount of time.

The performance parameters for comparison include End-to-End Traffic Delay (EETD), Round-Trip-Time (RTT), and Throughput [28][29][30]. The performance metrics used in all tests are of various packet sizes and various numbers of transmitting nodes in the ISP network.

The following subsections detailed the scenarios for each of these metrics used to evaluate and compare CHANC performance with other protocols.

4.1 End-to-End Traffic Delay (EETD)

The EETD was calculated and analyzed for all IPv4, IPv6, 6rd, D6across4, and CHANC based networks. **Fig. 5** shows the EETD with varying number of nodes in the network. **Fig. 5** also illustrates four different scenarios. Each scenario uses a certain packet size. The horizontal axis indicates the number of nodes in local access network used to simulate each network type. The vertical axis indicates the packet EED in terms of second (sec). The EETD for all nodes in each network type was calculated at a specified packet as the $\text{EETD}_{\text{NETWORK}}$ for all nodes. The $\text{EETD}_{\text{NETWORK}}$ for the ISP was calculated as given in Eq. (1) and Eq. (2).

$$EETD_{NODE} = \frac{\sum_{i=1}^{N} T_{end(i)} - T_{start(i)}}{N_{\mathrm{Rec}}} \tag{1}$$

$$EETD_{NETWORK} = \frac{\sum_{j=1}^{k} EETD_{NODE}(j)}{k} \tag{2}$$

Where,
$EETD_{NODE}$: EETD value for a certain node.
$EETD_{NETWORK}$: EETD value for the whole network at a specified packet size.
N_{Rec}: Total number of packets received at destination host.
T_{start}: Time of packet at source node.
T_{end}: Time of packet at destination node.
k: Total number of nodes in the access network.

In computing EETD and Round-Trip-Time (see next section), we must take the following factors into consideration [31]:

1. Processing Time: it is the overhead generated by sender and receiver at the time of submission and reception. If one of them does encapsulation or decapsulation, this time must be added to the submission/reception time.
2. Transmission time: it is the required time the sending node takes to transmit all bits to the medium.

Evaluated as $\dfrac{Transmitted\_bits}{Badwidth}$

3. Propagation time: it is the required time that takes the bit to traverse the medium.

Evaluated as $\dfrac{Medium\_length}{propagation\_speed}$

The propagation speed is affected if there is encapsulation, decapsulation, or mapping in the way between sending and receiving nodes. Hence, the propagation time will be effected as well.

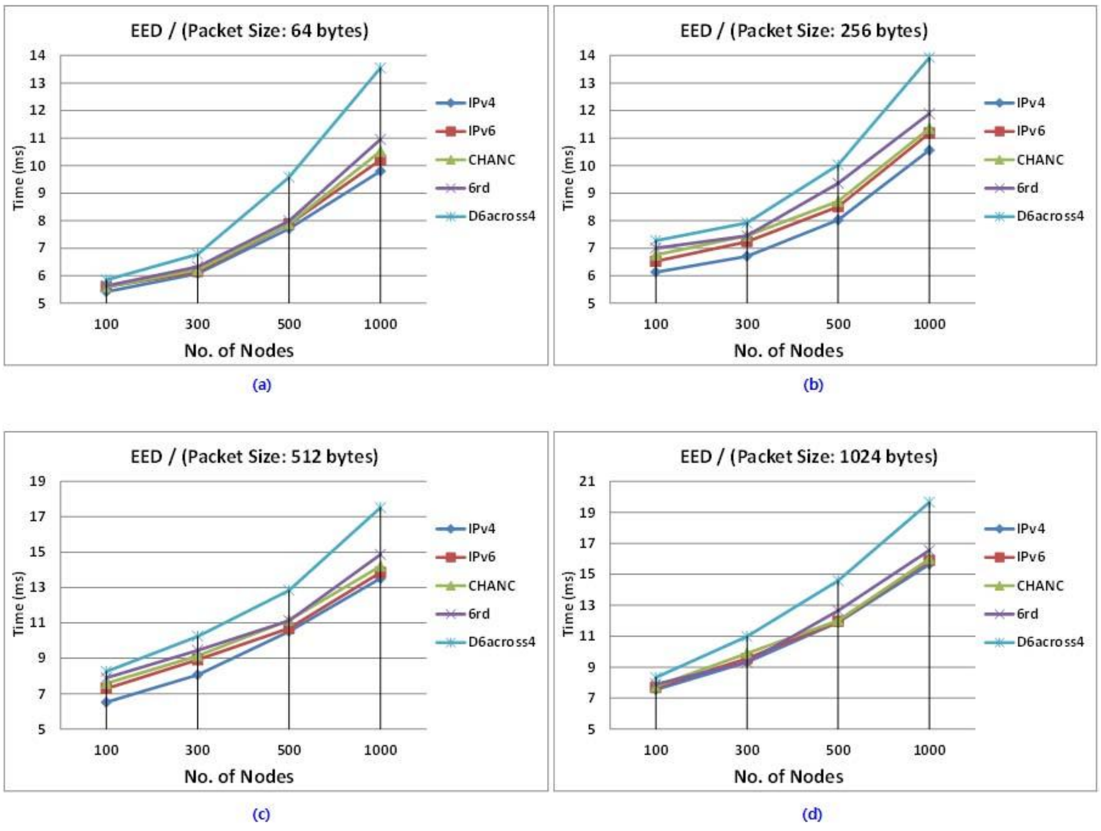

Fig. 5. End-to-End Delay Analysis

Referring to all EED analysis figures (A, B, C, and D), it could be noticed that the EED increased significantly with the increase in the packet size. Bigger packet size means more time consumed in delivering the packet to its destination due to the higher payload, which results in increasing the queuing delay for each transmitted packet. In our simulation, we tested two other factors that affected the EETD$_{NETWORK}$. Such factors are the number of nodes in the access network, and whether the protocol is stateful or stateless. The increasing number of nodes in the access network leads to increasing the probability of collision, contention, and hence increasing the average EED. Since D6across6 is a stateful operation, all subfigures of **Fig. 5** show that D6across4 traffic consumes the maximum time. D6across4 uses a mapping table to record the addressing information. Furthermore, D6across4 uses decapsulation/encapsulation for each outgoing/incoming packet which produces additional overhead. We noticed the significant increase in the $\text{EETD}_{\text{NETWORK}}$ for D6across4 following the increasing number of nodes in the network. The increasing number of nodes in the network makes the size of mapping table approximately large, and hence increases the queuing delay.

The results showed that the CHANC protocol gives the shortest delay in comparison to 6rd and D6across4 protocols. Like D6across4, 6rd uses encapsulation/decapsulation operations in the communication. Because 6rd is a stateless operation, only it consumes a short time in packet encapsulation/decapsulation which CHANC does not use. **Table 2** lists the percentage increase of EETD values in all protocols over IPv4 EETD values when packet size equals to 64 and 1024 bytes. It is important to highlight here that increasing packet sizes in CHANC, 6rd, and D6across4 leads to decrease the EETD, which result in better performance. The increasing packet sizes results in decreasing the number of transmitted packets in the network (i.e. for the same number of nodes). Thus, this leads to decrease the overhead that was generated in encapsulating/decapsulating packets and/or accessing the mapping table.

Table 2: percentage increase of EETD values over IPv4 EETD values.

	IPv6		**CHANC**		**6rd**		**D6across4**	
Nodes	64 Bytes	1024 Bytes	64 Bytes	1024 Bytes	64 Bytes	1024 Bytes	64 Bytes	1024 Bytes
100	1.16%	1.84%	2.26%	1.85%	3.71%	2.21%	7.92%	3.24%
300	2.13%	2.78%	2.29%	1.93%	3.89%	2.58%	11.59%	7.847%
500	3.12%	3.70%	3.23%	2.35%	4.19%	5.87%	24.24%	13.33%
1000	4.08%	4.41%	7.35%	3.45%	11.73%	9.72%	38.22%	20.82%

4.2 Round-Trip-Time (RTT)

RTT is the time that the packet takes to be delivered to its destination host, plus the time for its acknowledgment to be received at the source host. Traditionally, RTT value is very similar in both IPv4 and IPv6 based networks [32]. However, RTT for IPv6 networks was higher than RTT in IPv4 networks when the same packet size was used. Normally this happened because of the larger header size in IPv6 networks (i.e. 40 bytes) in comparison to header size in IPv4 networks (i.e. 20 bytes) which produces extra overhead. In this context, the RTT was tested to compare the delay in the ISP's border router (i.e. CRS in CHANC; TS in D6across4; Border relay Router in 6rd). This comparison would help to measure the amount of overhead generated at each of these border routers. Hence, we can indicate the QoS of the network type. Different tests were conducted to measure RTT with varying number of nodes in access networks and varying packet sizes. The total RTT of the network was measured as the average RTT for a certain number of nodes at a specified packet size. The average RTT was calculated as given in Eq. (3) and Eq. (4). It was calculated by transmitting N packets for each node individually. Then the reported value of $\text{RTT}_{\text{NETWORK}}$ is the average of all recorded values of all RTT_{NODE} at a specified packet size.

$$RTT_{NODE} = \frac{\sum_{i=1}^{N}(Tr_i - Ts_i)}{N} \tag{3}$$

$$RTT_{NETWORK} = \frac{\sum_{j=1}^{k} RTT_{NODE}(j)}{k} \tag{4}$$

Where,

RTT_{NODE}: RTT value for a certain node.
$RTT_{NETWORK}$: RTT value for the whole network at a specified packet size.
N: number of transmitted packets.
Tr_i: time when packet 'i' received at source host.
Ts_i: time when packet 'i' transmitted from source host.
k: total number of nodes in the access network.

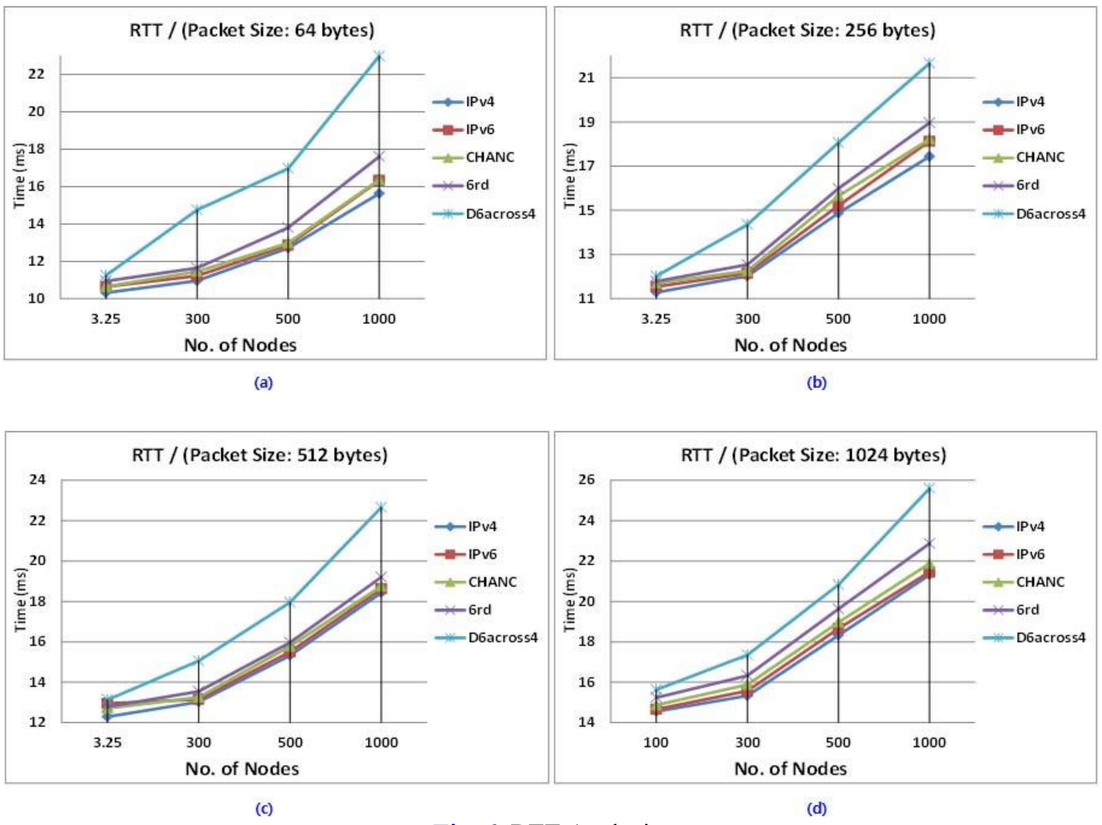

Fig. 6. RTT Analysis

Fig. 6 shows the $\text{RTT}_{\text{NETWORK}}$ results of all network types with varying numbers of nodes and packet sizes. Usually, larger packet sizes lead to extra overhead, and hence larger delay. However, all RTT figures are smaller than 19.2 ms for all network types except in D6across4 based networks. Also the maximum values for RTT in all figures belong to D6across4 protocol. We can conclude that larger packets with large number of nodes in the D6across4 based network generate the highest RTT values in comparison to other protocols. Because D6across4 protocol is a stateful protocol, it generates a lot of overhead in accessing large mapping table at every received packet. Moreover, another overhead was generated while encapsulating/decapsulating packets. These entire factors make D6across4 has the highest RTT values in comparison to other protocols. The gap between D6across4 and other protocols is reduced when considering large packet sizes (i.e. 1024). **Fig. 6 (d)** shows the smallest gap between RTT values in D6across4 and other protocols. The 6rd protocol on the other hand is a stateless protocol. Therefore, there is no mapping table used. But a small overhead was generated in encapsulation/decapsulation packets only. As CHANC is a stateless operation protocol and uses application level tunneling technique, all subfigures in **Fig. 6** showed that there is a large variance in RTT values between CHANC and D6across4. The stateless operation and application level tunneling make CHANC produce a very small overhead in comparison to D6across4 and 6rd protocols. The RTT values of CHANC are pretty near to both IPv4 and IPv6 RTT values. **Table 3** lists the percentage increase of RTT values in all protocols over IPv4 RRT values when packet size equals to 64 and 1024 bytes. Like EETD, increasing packet sizes in CHANC, 6rd, and D6across4 leads to decrease the RTT, which result in better performance. Increasing packet sizes leads to decrease the number of transmitted packets on the network (i.e. for the same number of nodes).

As a result, this leads to decreasing the overhead that was generated in encapsulating/decapsulating packets and/or accessing the mapping table. As shown in table 3, the smallest percentage value is 0.73%, which is related to IPv6 protocol when packet size = 1024 bytes. Moreover, the largest percentage value is 47.03%, which is related to D6across4 protocol when packet size = 64 bytes.

Table 3: percentage increase of RTT values over IPv4 RTT values.

	IPv6		**CHANC**		**6rd**		**D6across4**	
Nodes	64 Bytes	1024 Bytes	64 Bytes	1024 Bytes	64 Bytes	1024 Bytes	64 Bytes	1024 Bytes
100	2.30%	0.73%	2..37%	1.08%	6.08%	4.73%	9.01%	7.45%
300	3.25%	1.59%	3.55%	1.88%	6.93%	6.53%	23.60%	12.20%
500	4.08%	1.82%	4.78%	2.14%	8.60%	7.10%	33.5%	17.64%
1000	6.12%	2.04%	6.68%	2.43%	12.64%	2.98%	47.03%	23.06%

4.3 Throughput

Throughput can be defined as the measure of how much actual data can be transmitted over communication path per unit of time.

In our simulation, Throughput was measured for 6rd, D6across4, CHANC, IPv6, and IPv4 based networks with different numbers of concurrent transmitting nodes (100, 300, 500, 1000 nodes) and packet's sizes (64, 256, 512, 1024 bytes). In each case the Throughput was measured as the function of number of nodes at a specific packet size. The total time used to simulate each case is 100 seconds. The total Throughput was measured as the average Throughput for a certain number of nodes at a specified packet size. The average Throughput ($\text{THROUGHPUT}_{\text{NETWORK}}$) was calculated as given in Eq. (5) and Eq. (6).

$$THROUGHPUT_{NODE} = \frac{\sum_{i=1}^{20} \frac{N_{\mathrm{Rec}(i)} \times P_{size(i)} \times 8}{\left(T_{end(i)} - T_{start(i)}\right) \times 1 \times 10^{6}}}{20} \tag{5}$$

$$THROUGHPUT_{NETWORK} = \frac{\sum_{j=1}^{k} THROUGHPUT_{NODE}(j)}{k} \tag{6}$$

Where,

$THROUGHPUT_{NODE}$: throughput value for a certain node

$THROUGHPUT_{NETWORK}$: throughput value for the whole network at a specified 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 nodes in the access network.

For accuracy, the Throughput was calculated 20 times for each node individually. The reported value for each node is the average of all recorded values at a specified packet size.

Fig. 7 investigated the comparison of average Throughput for IPv6, IPv4, 6rd, D6across4, and CHANC based networks with various packet sizes. The D6across6 protocol has showed a reduced significant improvement average Throughput compared to 6rd and CHANC protocols. The simulation results showed that CHANC has achieved 5.95% and 2.61% higher average Throughput than 6rd and D6across4 respectively when packet size is 1024 bytes and total number of nodes is 100.

The behavior of the average Throughput for single TCP stream traffic with various packet sizes and different number of nodes were evaluated. **Fig. 7** shows the measured average Throughput as a function of number of nodes, it can be seen both 6rd and CHANC achieved a good average Throughput in comparison to IPv6 and IPv4 protocols. But, D6across4 showed a reduced average Throughput at all packet sizes. In all conducted scenarios, we have noticed that when the number of nodes in the network increased, the average Throughput decreased. More nodes in the network lead to increasing the probability of collision, contention, and delay. Therefore, this will negatively effect on the overall average Throughput. However, the average Throughput for 6rd and CHANC protocols was decreased a little bit in comparison to D6across4 protocol when considering a larger number of nodes in the network. This could be attributed to the stateless operation of 6rd and CHANC protocols. The D6caross4 average Throughput cannot go up further in all the scenarios. This was due to the encapsulation/decapsulation for every incoming/outgoing packet. Moreover, D6across4 protocol maintains a mapping table to record addressing information for each communication session. A considerable overhead was generated at the tunnel server when a large number of nodes were communicating simultaneously. The tunnel server will access mapping table at the reception of every packet to locate or add addressing information. Also the D6across4 tunnel server will encapsulate/decapsulate incoming/outgoing packets. These things make D6across4 average Throughput decreasing significantly at the large number of nodes in the network. Based on these results, D6across4 can be used when number of nodes < 300 with packet size is greater than or equal to 1024 bytes. **Table 4** lists throughput values for all protocols with packet sizes 64 and 1024 bytes.

We noticed from **Fig.** 7(A) that the IPv4 average Throughput is highest amongst other protocols. This was due to the small header size (20 bytes) in comparison to IPv6 header (40 bytes). But when considering larger packet sizes (i.e. Figures (B, C, D)) the IPv6 average Throughput achieved the highest throughput among others. In general, in all scenarios CHANC average Throughput achieved better than 6rd average Throughput and it was pretty nearly to IPv6 average Throughput. As explained earlier, CHANC posts IPv6 packets over HTTP protocol without generating extra overhead in encapsulation or decapsulation. This makes CHANC achieve better average Throughput among other tunneling protocols.

Table 4: throughput values for all protocols with packet sizes 64 and 1024 bytes

	IPv4		**IPv6**		**CHANC**		**6rd**		**D6across4**	
Nodes	64 B	1024 B	64 B	1024 B	64 B	1024 B	64 B	1024 B	64 B	1024 B
100	14.98	74.34	14.83	74.46	14.87	74.27	14.26	72.33	14.15	69.84
300	12.84	70.28	12.04	70.90	11.89	70.03	11.56	68.57	10.41	63.55
500	9.67	60.30	9.01	60.92	8.52	60.25	9.09	59.05	5.95	49.18
1000	5.32	41.46	4.8	41.81	4.66	41.13	4.35	38.22	0.55	29.79

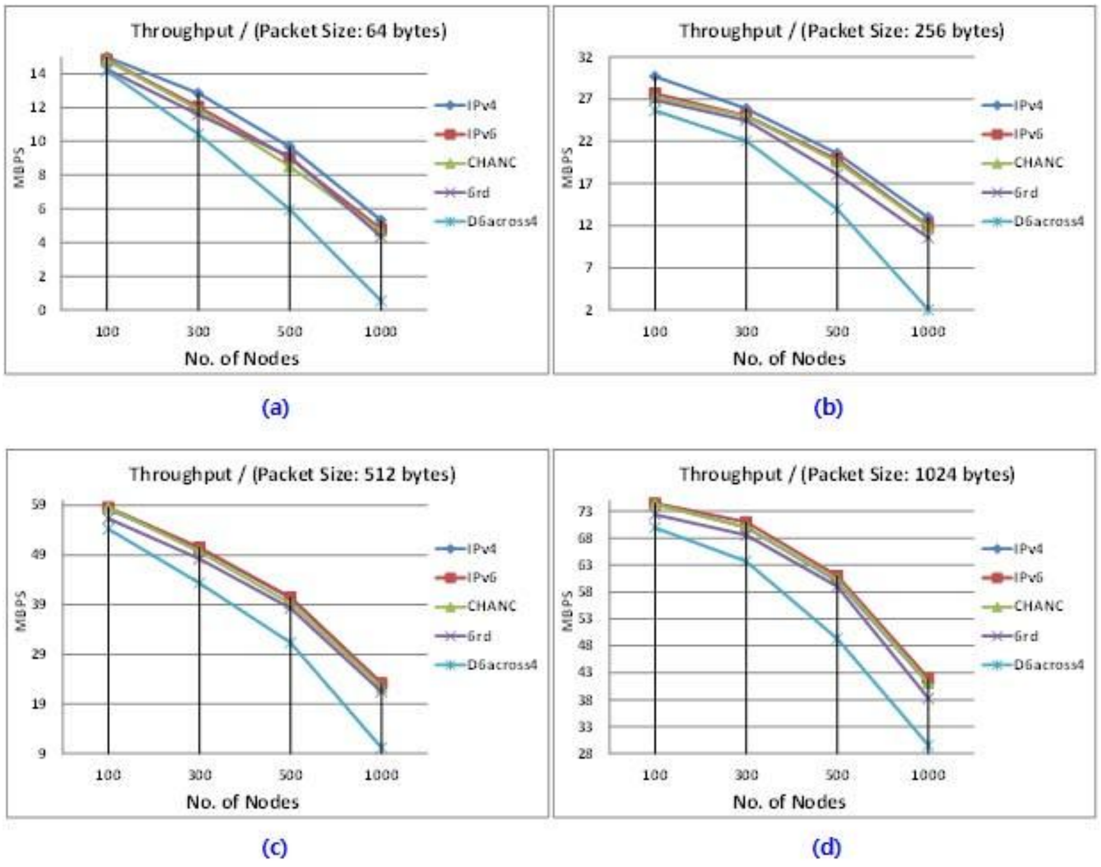

Fig. 7. Throughput Analysis

5. Conclusions and Future Work

This paper introduced a new protocol called CHANC. It helps in rapid deployment of IPv6 service across ISPs' IPv4-only access networks. CHANC is developed to help ISPs to continue providing their IPv4 service alongside IPv6 service. CHANC constitutes an optimal solution when IPv4 address space is completely exhausted, thus new launched ISPs can provide IPv6 service to their customers across IPv4-only access network. The key aspects of CHANC are: Stateless operation, automatic host configurations, no CE configurations required, uses application-level tunneling, its traffic can traverse firewalls, economical solution, and its performance is nearly equal to both IPv4 and IPv6 based networks. Instead of the traditional method of tunneling IPv6 packets in IPv4 packets, CHANC uses HTTP protocol to post IPv6 traffic over IPv4-only access networks. The simulation results showed that CHANC performance parameters are better than all current tunneling protocols and it is nearly equal to both IPv6 and IPv4 based networks. CHANC has showed a stable, reliable, and efficient performance especially when a large number of nodes are used in the IPv4-only access network.

For future research, the following improvements to CHANC could be proposed:

- To avoid bottlenecks and single point of failures at CRS, ISPs can configure their network with multiple CRSs.
- Conduct tests to measure UDP traffic for all tunneling protocols (6rd, D6across4, and CHANC) with different number of nodes and packet sizes.

- Conduct tests to measure and compare tunnel setup delay and query delay parameters between all protocols.
- Use Internet Content Adaptation Protocol (ICAP) [33] instead of HTTP protocol and conduct tests to compare between CHANC performance with HTTP and ICAP protocols.

6. Acknowledgment

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

References

[1] Postel J., "Internet Protocol," *Internet Engineering Task Force RFC 791*, 1981. Article (CrossRef Link).
[2] S. Kent, K. Seo, "Security Architecture for the Internet Protocol," *Internet Engineering Task Force RFC 4301*, 2005. Article (CrossRef Link).
[3] Srisuresh, P., K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)," *Internet Engineering Task Force RFC 3022*, 2001. Article (CrossRef Link).
[4] V. Fuller, T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan," *Internet Engineering Task Force RFC 4632*, 2006. Article (CrossRef Link).
[5] Deering, S., R., Hinden, "IP Version 6 Addressing Architecture," *Internet Engineering Task Force RFC 4291*, 2006. Article (CrossRef Link).
[6] J. Davies, "Understanding IPv6," *Microsoft Press*, 2003.
[7] B. Forouzan,"TCP/IP Protocol Suite," *McGraw-Hill*, 2003.
[8] J. F. Kurose, K. W. Ross, "Computer Networking A Top-Down Approach Featuring the Internet," *Pearson Education*, New York, 2009.
[9] N. Oliver, V. Oliver; "Computer Networks Principles, Technologies and Protocols for Network Design," *John Wiley and Sons*, England, 2006.
[10] J. Chen, Y. Chang, C. Lin, "Performance Investigation of IPv4/IPv6 Transition Mechanisms," *Journal of Internet Technology*, vol. 5 no.2, pp. 163-170, 2004. Article (CrossRef Link).
[11] A. Hamarsheh, M. Goossens, R. Alasem, "Decoupling Application IPv4/IPv6 Operation from the Underlying IPv4/IPv6 Communication (DAC)," *American Journal of Scientific Research, Eurojournals Press*, no. 14, pp. 101-121, 2011. Article (CrossRef Link).
[12] S. Lagerholm, "Service Provider Transition to IPv6," *Secure64 Software Corporation-white paper*, 2011. Article (CrossRef Link).
[13] F. Baker, X. Li, C. Bao, K. Yin, "Framework for IPv4/IPv6 Translation," *Internet Engineering Task Force RFC 6144*, 2011. Article (CrossRef Link).
[14] M. Analoui and M. H. Rezvani, "A Framework for Resource Allocation in Multi-Service Multi-Rate Overlay Networks Based on Microeconomic Theory," *Journal of Network and Systems Management*, vol. 19, no. 2, pp. 178-208, 2011. Article (CrossRef Link).
[15] W. Townsley, O. Troan, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd) -- Protocol Specification," *Internet Engineering Task Force RFC 5969*, 2010. Article (CrossRef Link).
[16] R. Despres, "IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)," *Internet Engineering Task Force RFC 5569*, 2010. Article (CrossRef Link).
[17] A. Hamarsheh, M. Goossens, R. Alasem, "Deploying IPv6 Service Across Local IPv4 Access Networks," in *10th WSEAS International Conference on TELECOMMUNICATIONS and INFORMATICS (TELE-INFO '11)*, pp. 94-100, Lanzarote, Canary Islands, Spain, May 27-29, 2011. Article (CrossRef Link).

[18] Internet Architecture Board, "IAB Technical Comment on the Unique DNS Root," *Internet Engineering Task Force RFC 2826*, 2000. Article (CrossRef Link).
[19] R. Droms, J. Bound, B. Volz, T. Lemon, C. Perkins, M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," *Internet Engineering Task Force RFC 3315*, 2003. Article (CrossRef Link).
[20] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1," *Internet Engineering Task Force RFC 2616*, 1999. Article (CrossRef Link).
[21] Quin, L., "Extensible Markup Language (XML)," *W3C*, 2003.
[22] Bao C., Huitema C., Bagnulo M., Boucadair M., Li X., "IPv6 Addressing of IPv4/IPv6 Translators," *Internet Engineering Task Force RFC 6052*, 2010. Article (CrossRef Link).
[23] "DHCPv6 Linux," http://www.hycomat.co.uk/dhcp/.
[24] S. Thomson, C. Huitema, V. Ksinant, M. Souissi, "DNS Extensions to Support IP Version 6," *Internet Engineering Task Force RFC 3596*, 2003. Article (CrossRef Link).
[25] Isabel Dietrich, "OMNET++ Traffic Generator," http://www7.informatik.uni-erlangen.de/~isabel/omnet/modules/TrafGen/.
[26] "OMNeT++ Network Simulation Framework," http://www.omnetpp.org.
[27] "INET Framework", http://inet.omnetpp.org/.
[28] Dehury, R. K., "An analysis of QoS Traffic Engineer using CR-LDP Protocol over MPLS Networks," A Research study RSPR No. TC-01-5, Telecommunication Program, School of Advanced Technologies, *Asian Institute of Technology*, 2001.
[29] I. Raicu, S. Zeadally, "Evaluating IPv4 to IPv6 transition mechanisms", in *IEEE International Conference on telecommunications*, 2003. Article (CrossRef Link).
[30] R. Alja'afreh, J. Mellor, I. Awan, "Implementation of IPv4/IPv6 BDMS Translation Mechanism," in *Second UKSIM European Symposium on Computer Modeling and Simulation*, 2008. Article (CrossRef Link).
[31] E. Gamess, R. Suro´s, "An upper bound model for TCP and UDP throughput in IPv4 and IPv6," *Journal of Network and Computer Applications*, no. 31, pp. 585–602, 2008. Article (CrossRef Link).
[32] W. Shiau, Y. Li, H. Chao, P. Hsu, "Evaluating IPv6 on a large-scale network," *Computer Communications*, no. 16, pp. 3113-3121, 2006. Article (CrossRef Link).
[33] J. Elson, A. Cerpa, "Internet Content Adaptation Protocol (ICAP)," *Internet Engineering Task Force RFC 3507*, 2003. Article (CrossRef Link).

Ala Hamarsheh 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, Hamarsheh was working as a full time instructor at the Arab American University – Jenin, 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 PhD research focuses on proposing protocols 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.

Rafe Alasem received his BSc and MSc from Jordan University of Science and Technology (Jordan) in computer engineering (1992) and Communication and Electronics (2001) respectively. He received his PhD from the Department of Computing, University of Bradford, UK (2008). Currently, he is working as assistant professor at Computer Science Department, Imam Muhammad ibn Saud Islamic University, Saudi Arabia.