IPv6 (IP version 6) Essentials
Ch10 Aux:
Transition From IPv4 To IPv6

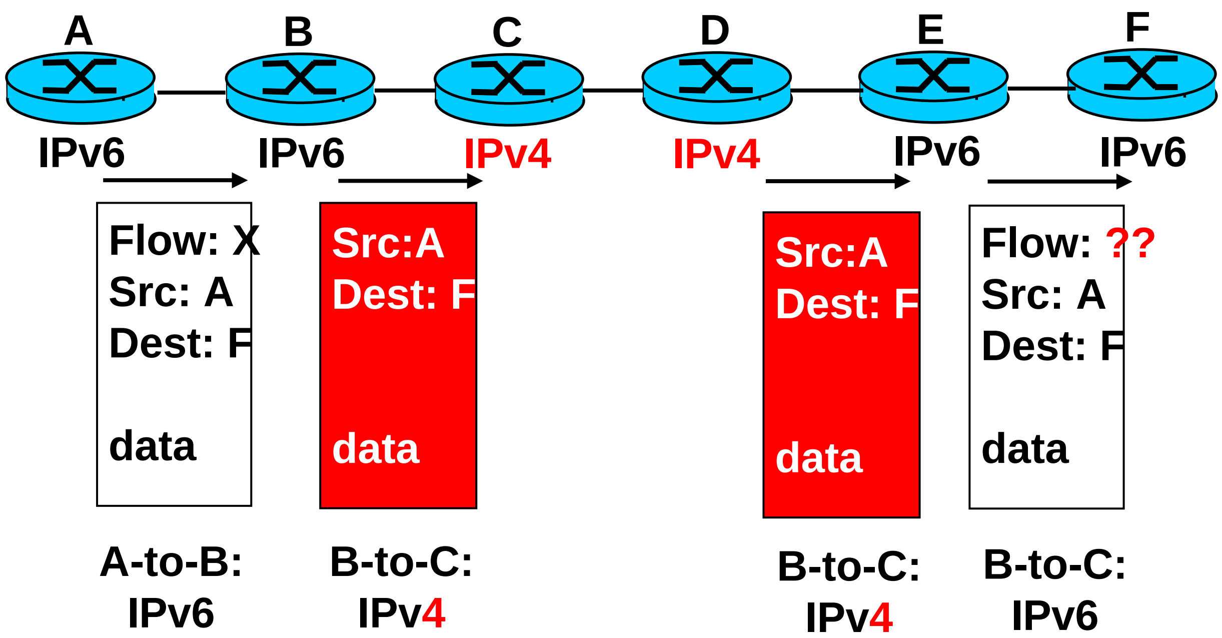
Louis Chuang
Fu Jen Catholic University
EE ENCL

Transition From IPv4 To IPv6 (1)

- IETF next generation transition (NGtrans): Proposed three transition mechanisms for different requirements (situations):
  - Dual Stack: some routers with dual stack (v6, and v4) can "translate" between formats (IPv4 and IPv6 coexist in the same nodes and networks).
    - Implement both IPv4 and IPv6 protocol stacks in the same equipment.
  - Tunneling: IPv6 packet is carried in IPv4 packet among IPv4 routers (IPv6 islands can connect with each other via IPv4 network).
    - Encapsulate/Decapsulate IPv6 packets.
  - Translator: executing addresses translation of IPv6 and IPv4 (IPv6 network can directly communicate with IPv4 network).
    - Perform packet format translation and addresses mapping method.

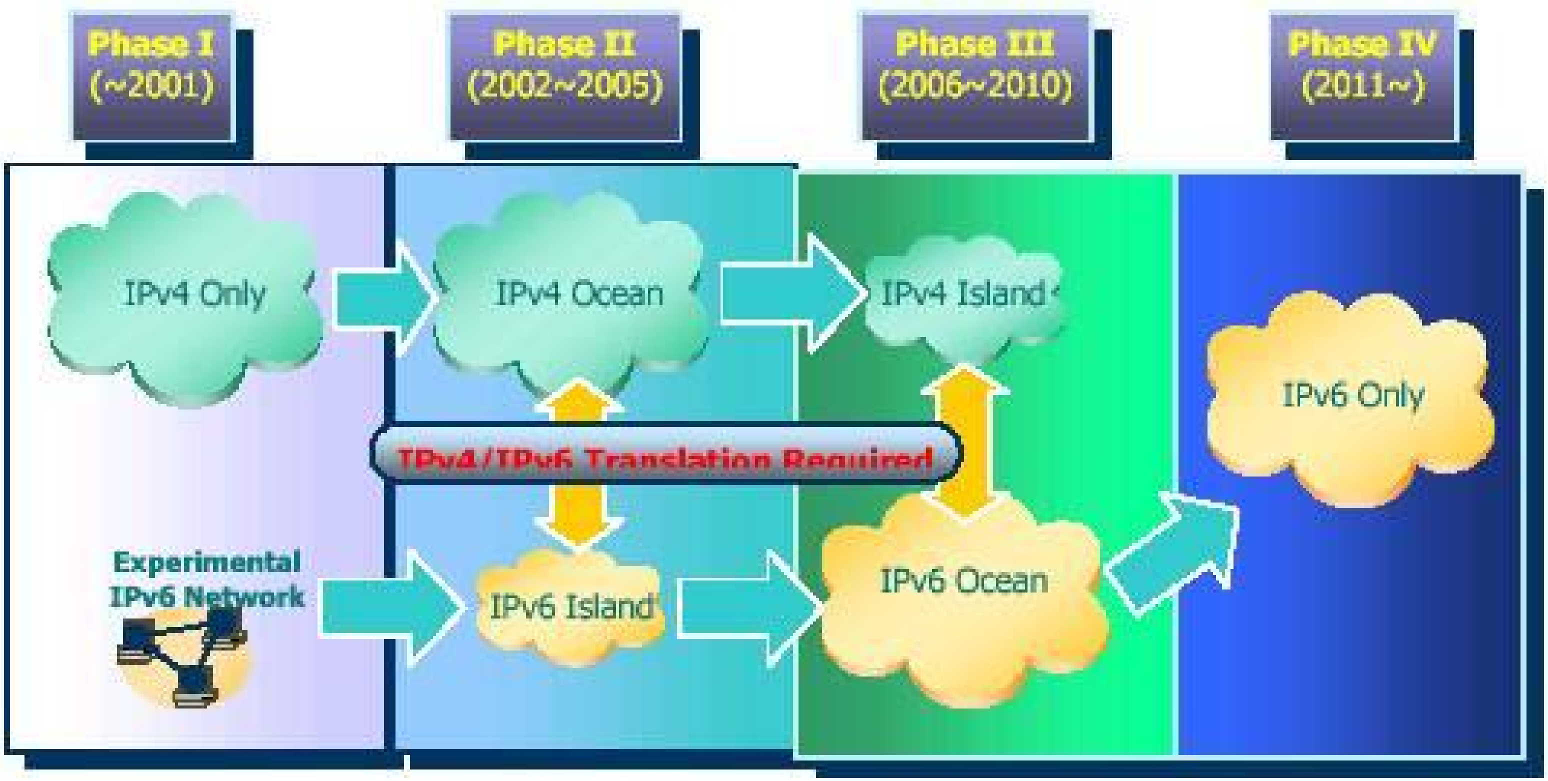
Transition From IPv4 To IPv6 (2)

- Why we need the transition from IPv4 to IPv6:
  - IPv4 packet format can not change directly to IPv6 packet format.



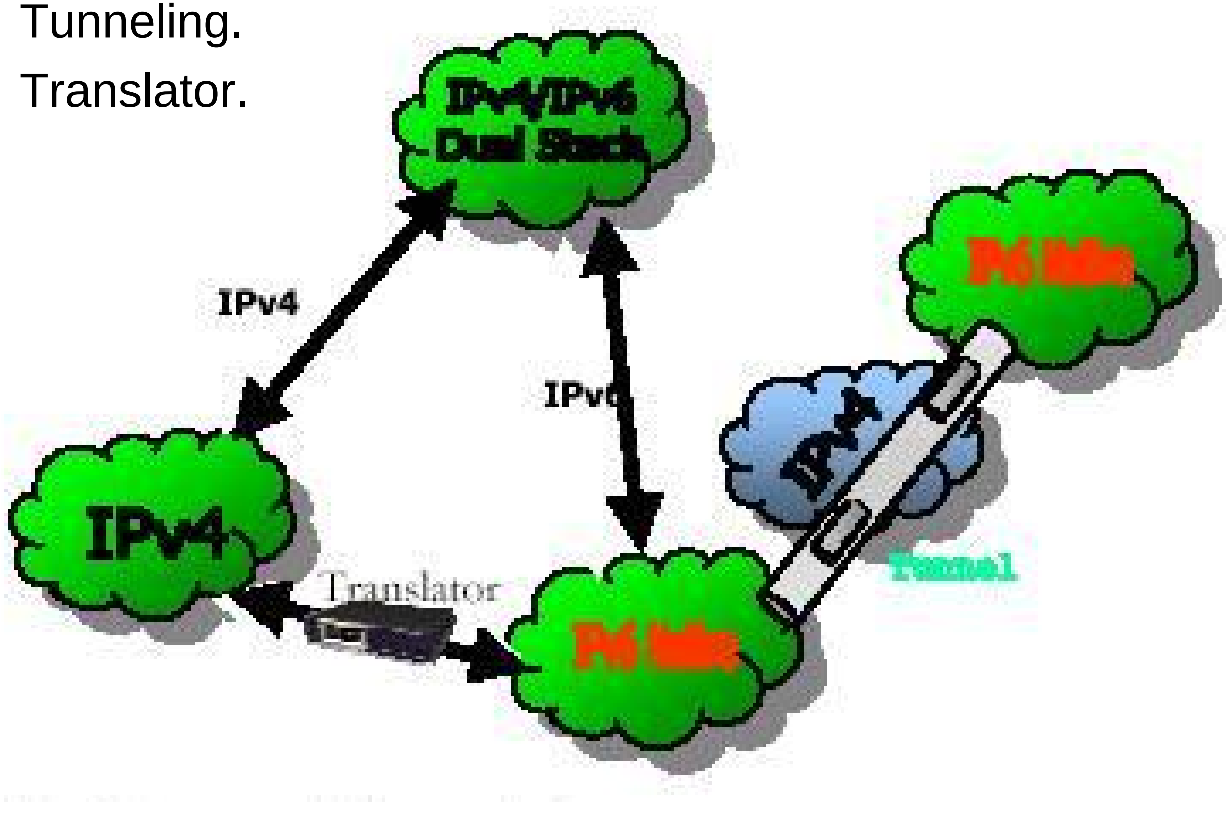
Transition From IPv4 To IPv6 (3)

- Transition schedule (from IPv4 to IPv6).



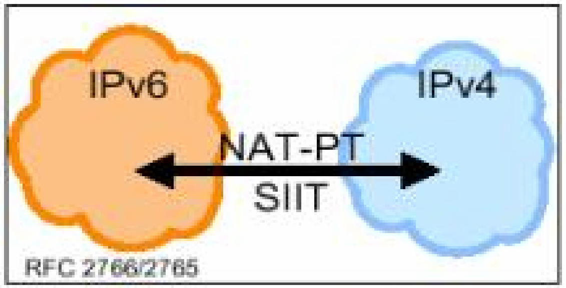
IPv4/IPv6 Transition Mechanisms (1)

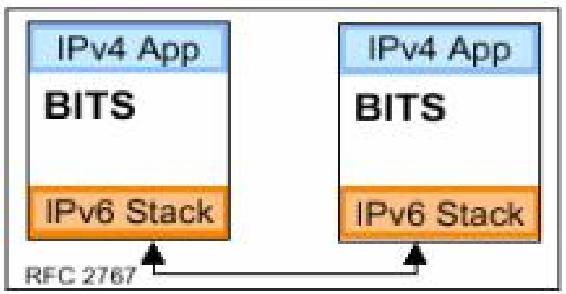
- Different mechanisms are for various connection conditions.
  - Dual Stack.
  - Tunneling.
  - Translator.

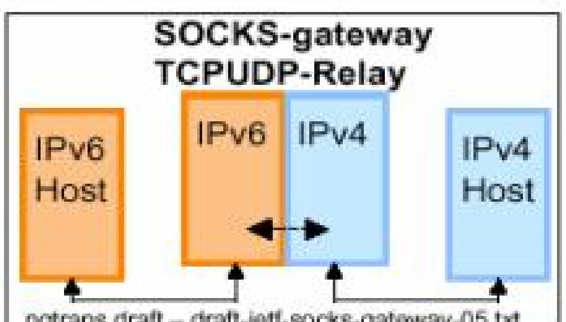


IPv4/IPv6 Transition Mechanisms (2)

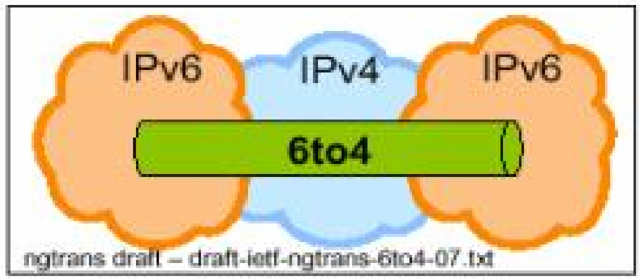
Translators

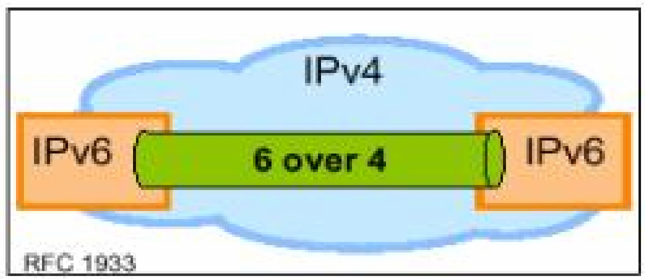


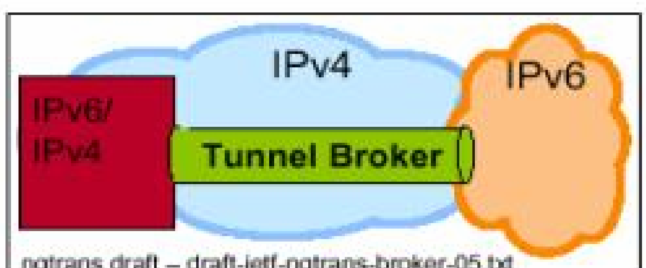




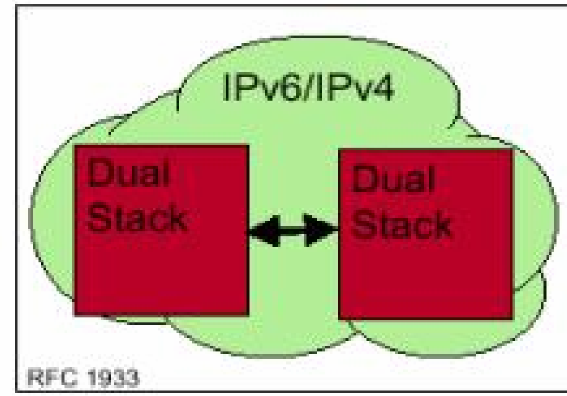
Tunneling

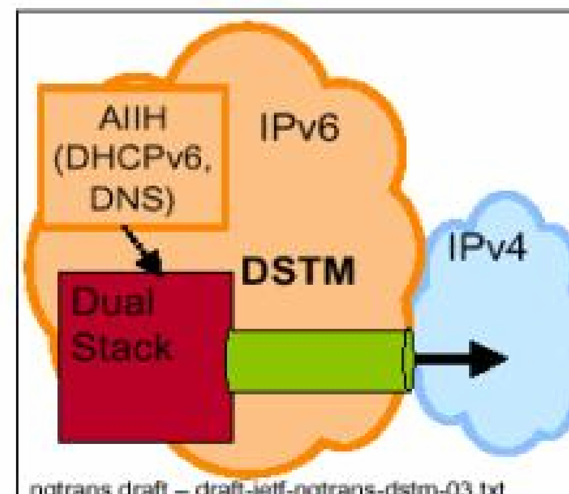






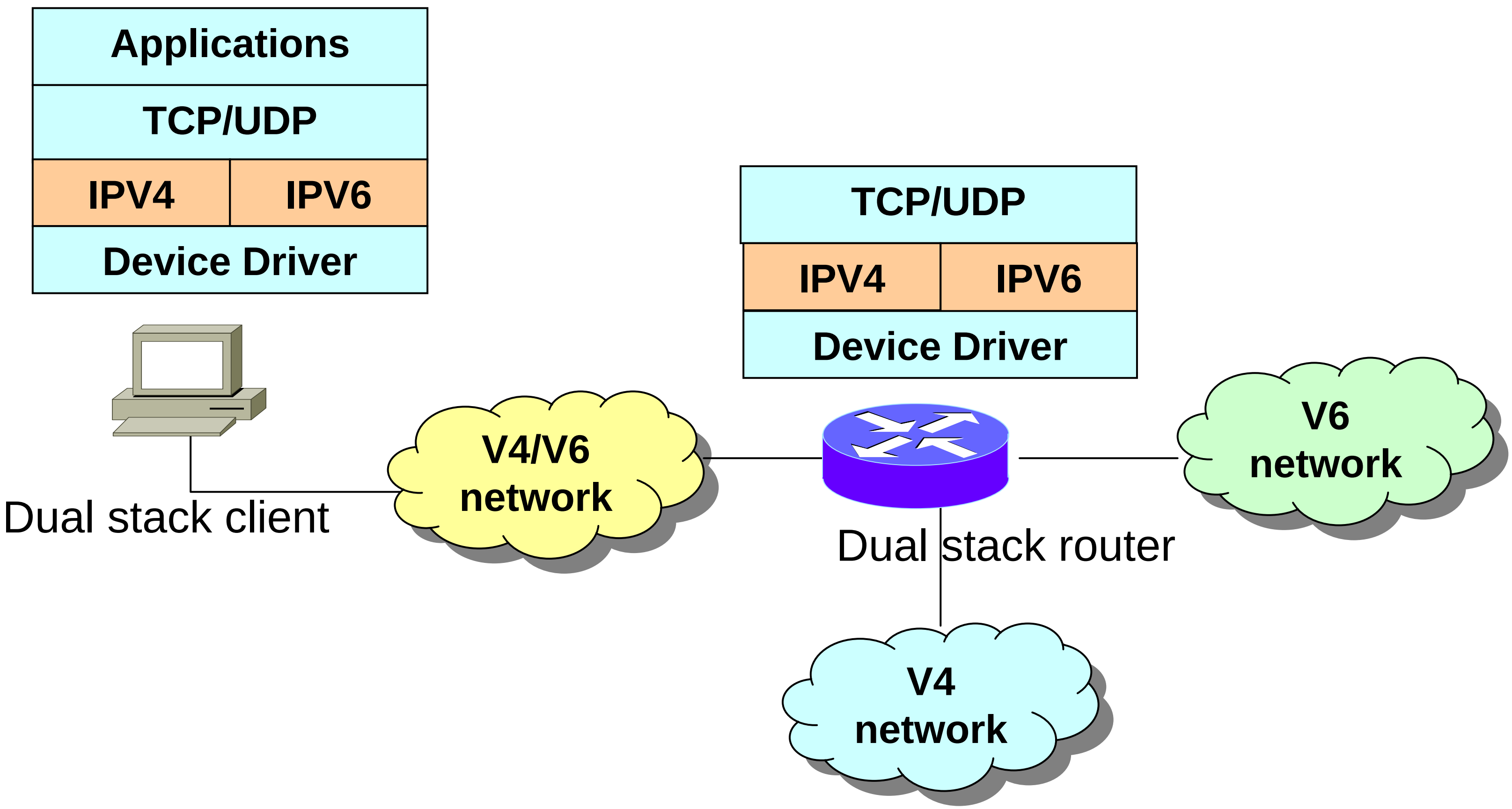
Dual Stack





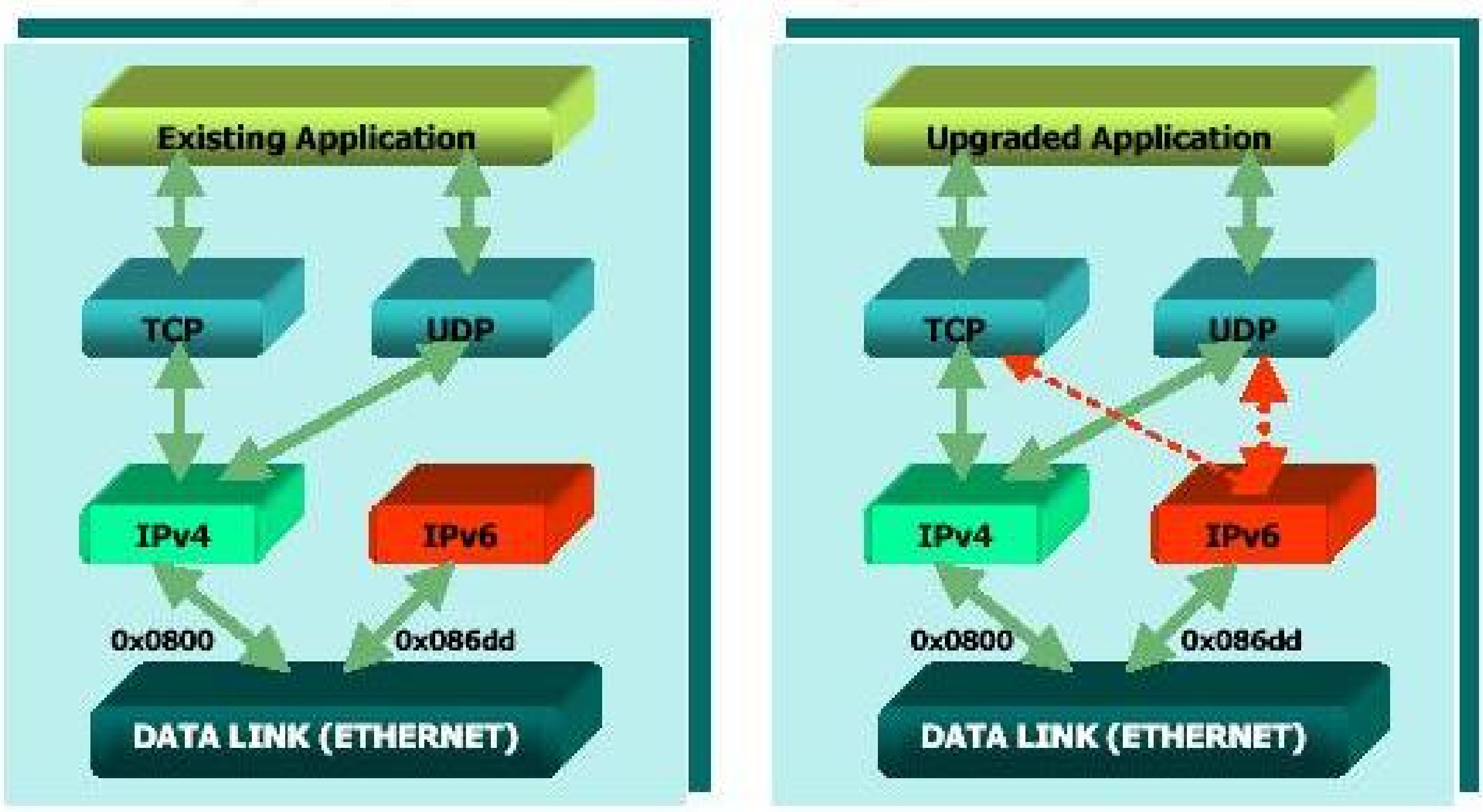
Dual Stack Mechanism (1)

- To install both IPv6 and IPv4 protocol stacks in a single node (client and router).



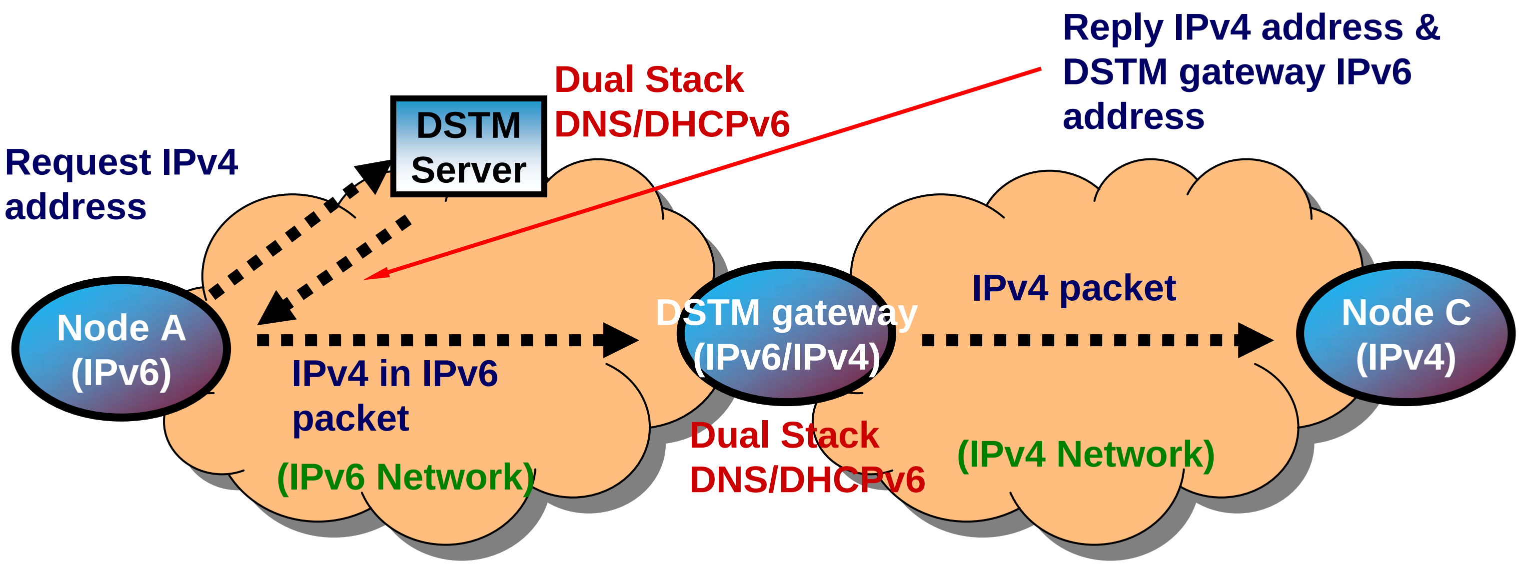
Dual Stack Mechanism (2)

- Using the same application, TCP, UDP, and data link layer protocols.



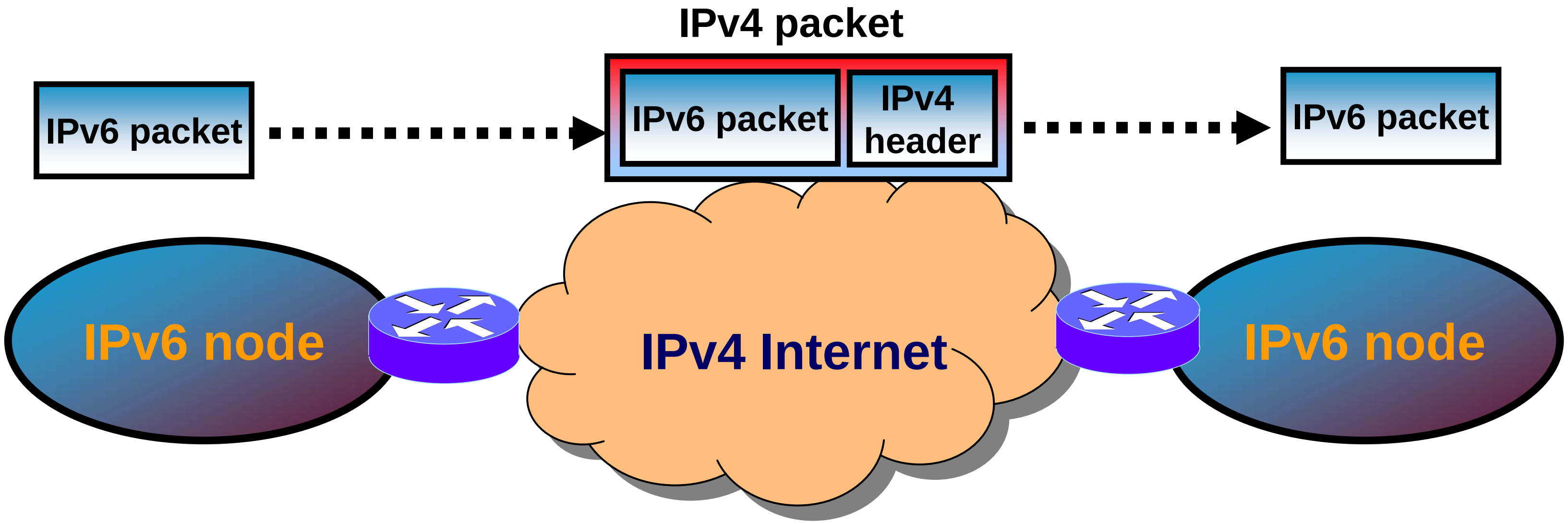
Dual Stack Mechanism (3)

- According to the destination node's IP version, select the protocol version (IPv4 or IPv6).
  - If it is IPv4,
    → Use IPv4 protocol stack.
  - If it is IPv6 or IPv6/IPv4(dual stack),
    → Use IPv6 protocol stack.
- Example: Dual Stack Transition Mechanism (DSTM)
  - Node A (IPv6) → node C (IPv4).
  - Requiring temporarily IPv4 address.

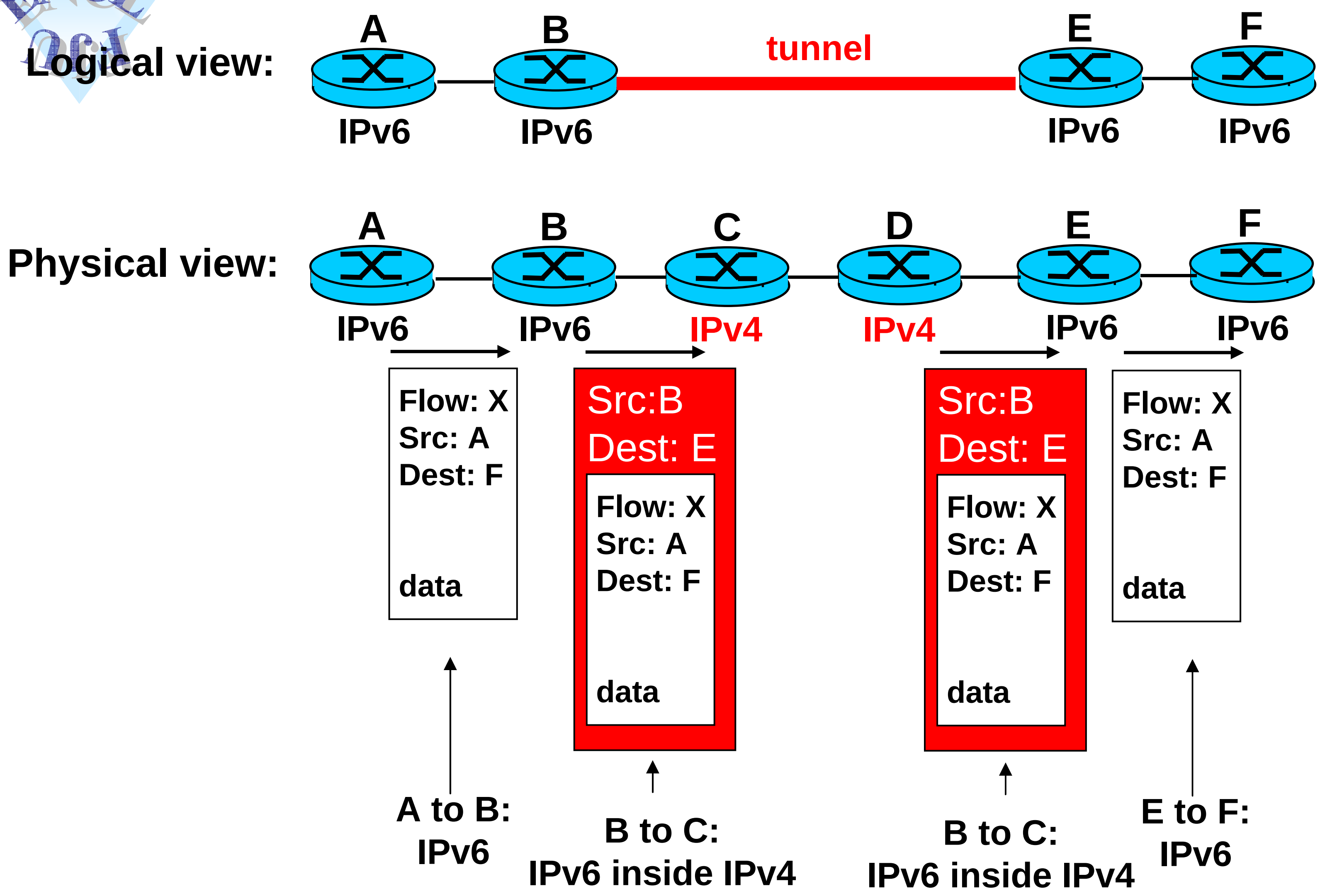


Tunneling Mechanism (1)

- Interconnect two IPv6 nodes using the virtual link over IPv4 network.
- IPv6 packet is encapsulated into IPv4 packet, using the IP encapsulation technique.
- Tunnel connection mechanisms:
  - Manually configured.
  - Semi-automated configured (e.g. Tunnel broker).
  - Automated configured (e.g. 6to4, 6over4, 6in4, 4to6, etc.).



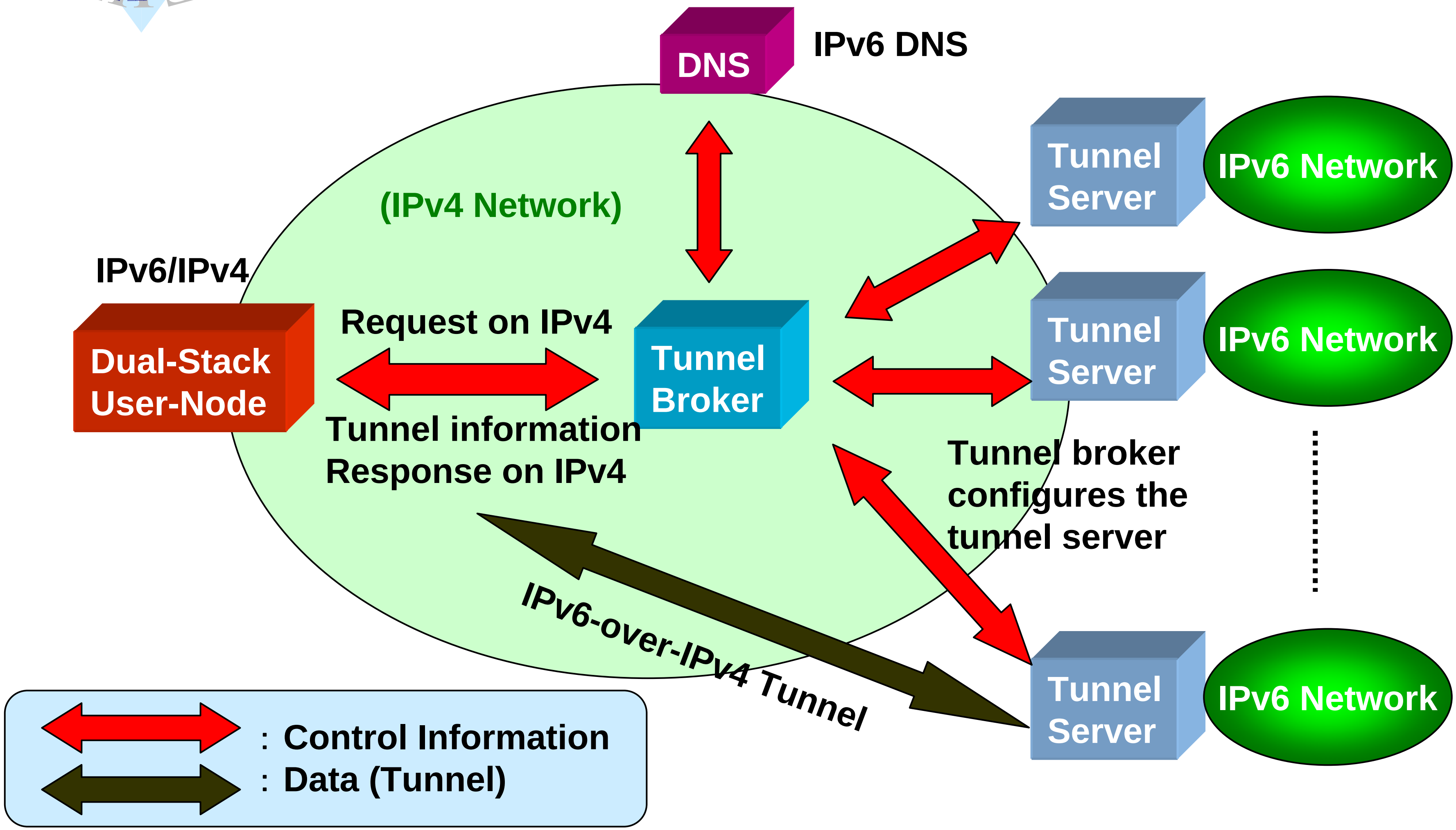
Tunneling Mechanism (2)



Tunneling Mechanism (3) (Tunnel Broker)

- Tunnel broker automatically manages tunnel requests coming from the users.
  - The tunnel broker fits well for small isolated IPv6 sites, especially isolated IPv6 hosts on the IPv4 Network.
- Client node (user) must be a dual stack node (IPv4/IPv6 node).
- The client IPv4 address must be globally routable (no NAT).

Tunneling Mechanism (4)
(Tunnel Broker)



Tunneling Mechanism (5) (6to4 Tunnel)

- 6to4:
  - Automatic p2p tunnel over IPv4 cloud (i.e., IPv6 packet in IPv4) .
  - Special IPv6 address:
    - IPv6 address contains: 6to4 prefix (2002::/16) and IPv4 address.
    - A global IPv4 address (the output port v4 address of 6to4 outer).
    - 6to4 network has 2002:v4Addr::/48.
    - Interface ID: can be the EUI-64 (encoded MAC address) or random etc.
  - Protocol ID field in IPv4 header is set to "41".

IPv6 Address Format of 6to4 Tunnel (128 bits)

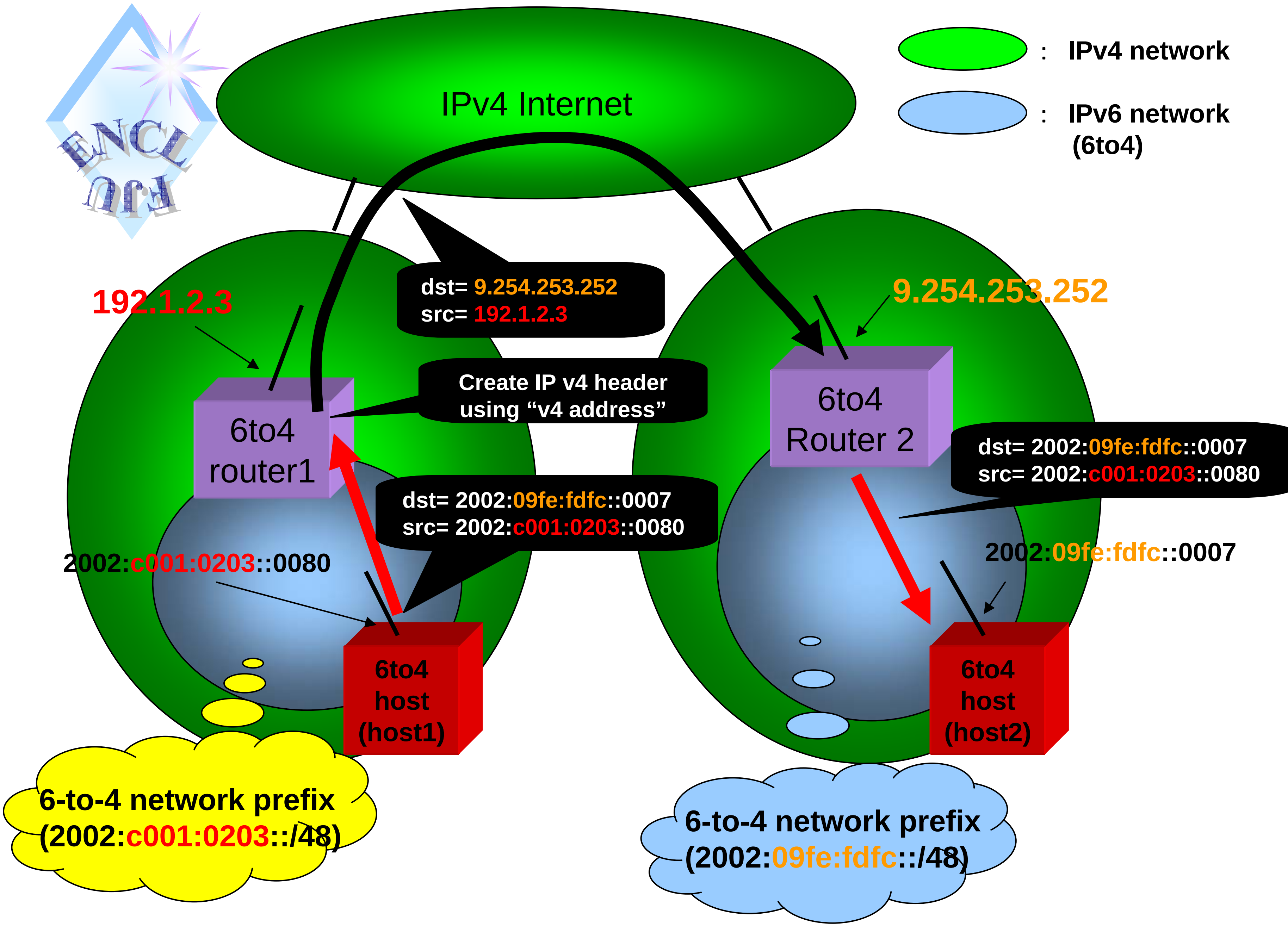
FP 001	TLA 0x0002	NLA V4 address	SLA ID	Interface ID	
3	13	32	16	64	(bits)

6to4 prefix:
Format Prefix (FP)+TLA
=2002 (16 bits)

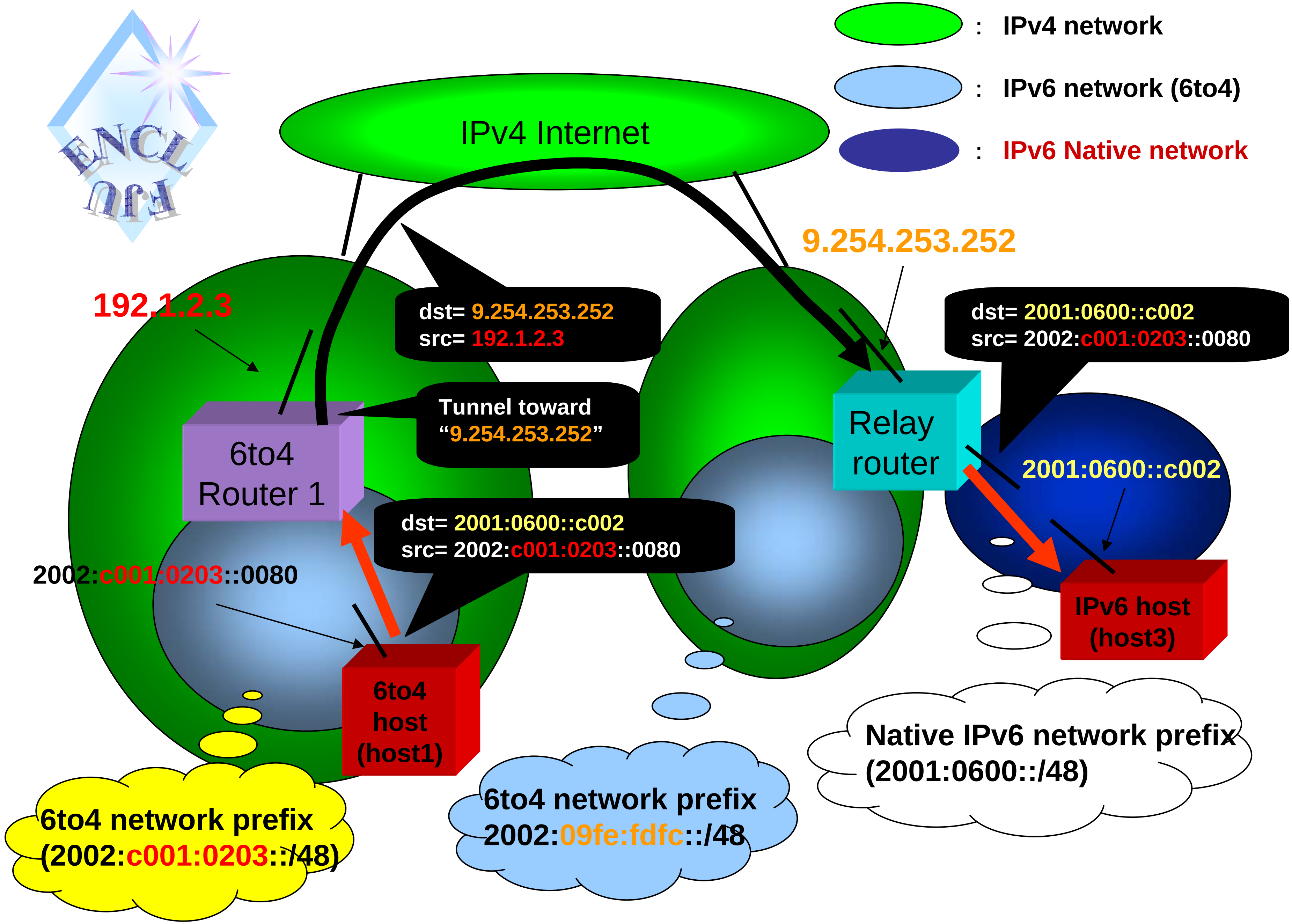
TLA: top-level aggregation identifier.
NLA: next-level aggregation identifier.
SLA: site-level aggregation identifier.

Tunneling Mechanism (6) (6to4 Tunnel)

- EUI-64 (Extended Unique Identifier):
  - Defined by IEEE (coded MAC address).
  - 6 bytes (48 bits) MAC address -> 8 bytes (64 bits) Interface ID.
    - Step1: insert 0xFFFE into a MAC address.
    - Step2: the second last bit of the first byte is inverted.
  - EX) MAC address is 00 48 54 86 D3 29.
    - Step1: 00 48 54 FF FE 86 D3 29.
    - Step2: the first byte 00 is: 0000 0000 (binary).
      invert the second last bit: 0000 0010 -> 02.
    - Step3: EUI-64: 02 48 54 FF FE 86 D3 29.
    - Step4: Interface ID is 0248:54FF:FE86:D329.



IPv4 Internet
: IPv4 network
: IPv6 network (6to4)
192.1.2.3
9.254.253.252
dst= 9.254.253.252
src= 192.1.2.3
Create IP v4 header using “v4 address”
6to4 router1
6to4 Router 2
dst= 2002:09fe:fdfc::0007
src= 2002:c001:0203::0080
dst= 2002:09fe:fdfc::0007
src= 2002:c001:0203::0080
2002:c001:0203::0080
2002:09fe:fdfc::0007
6to4 host (host1)
6to4 host (host2)
6-to-4 network prefix (2002:c001:0203::/48)
6-to-4 network prefix (2002:09fe:fdfc::/48)


: IPv4 network
: IPv6 network (6to4)
: IPv6 Native network
IPv4 Internet
9.254.253.252
192.1.2.3
dst= 9.254.253.252
src= 192.1.2.3
dst= 2001:0600::c002
src= 2002:c001:0203::0080
Tunnel toward
"9.254.253.252"
6to4
Router 1
Relay
router
2001:0600::c002
dst= 2001:0600::c002
src= 2002:c001:0203::0080
2002:c001:0203::0080
6to4
host
(host1)
IPv6 host
(host3)
Native IPv6 network prefix
(2001:0600::/48)
6to4 network prefix
2002:09fe:fdfc::/48
6to4 network prefix
(2002:c001:0203::/48)

Tunneling Mechanism (7) (6to4 Tunnel)

- Client user can automatically create a 6to4 IPv6 address via 6to4 router:
- Using two ICMPv6 message to get the information.
  - Router Solicitation (RS) message (type no. 133).
  - Router Advertisement (RA) message (type no. 134).

RS message Format (Multicast)

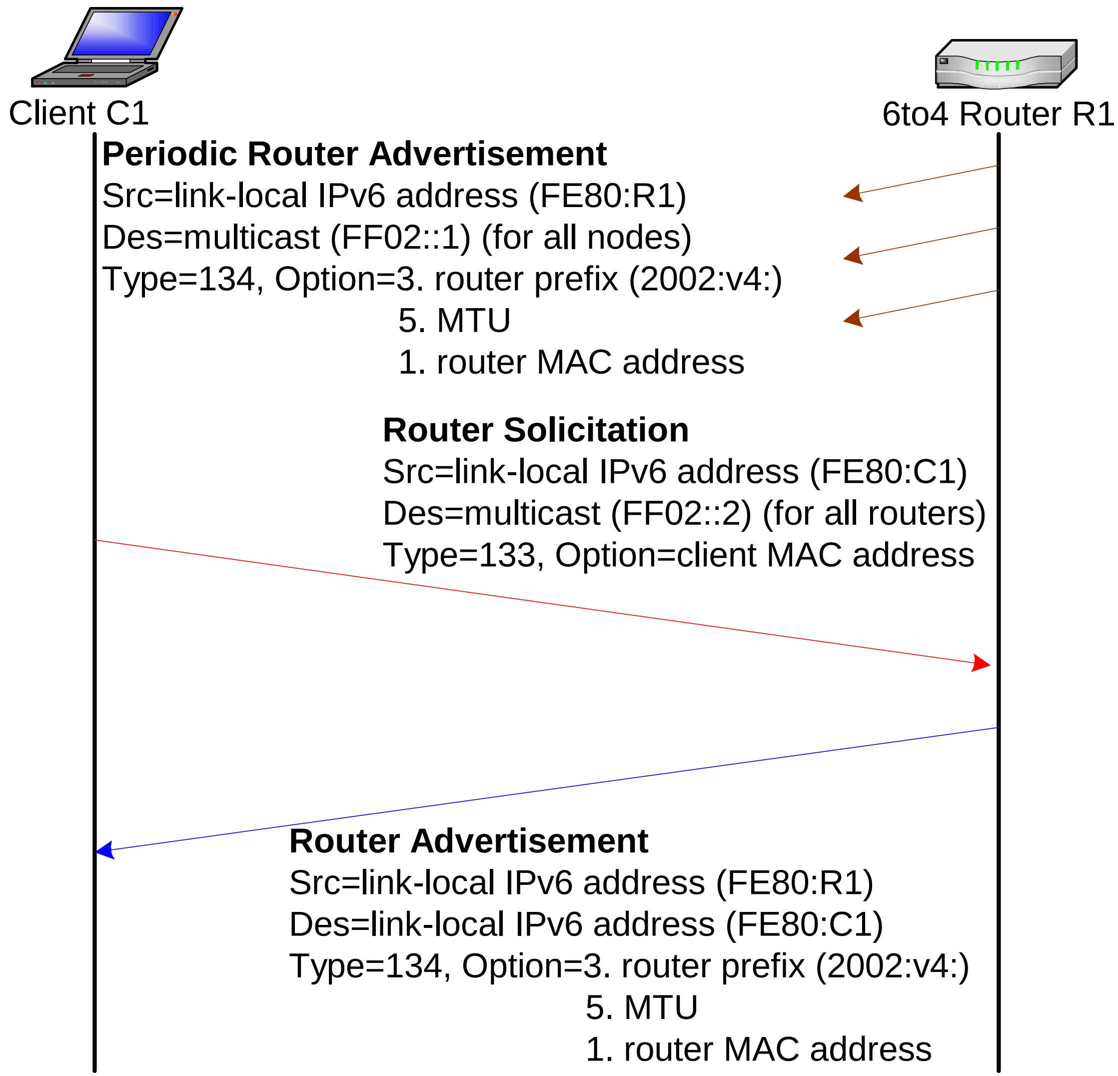
type 133	Code 0	Check Sum	Reserved 0			1. MAC Address
1	1	2	4 (bytes)	←	Option (variable)	→

RA message Format (Multicast/Unicast)

type 134	Code 0	Check Sum	Hop Count 0	State	Life Time	Reachable Time	Re-Tx			3. Perfix, 5. MTU 1. MAC Address
1	1	2	1	1	2	4	4 (bytes)	←	Option (variable)	→

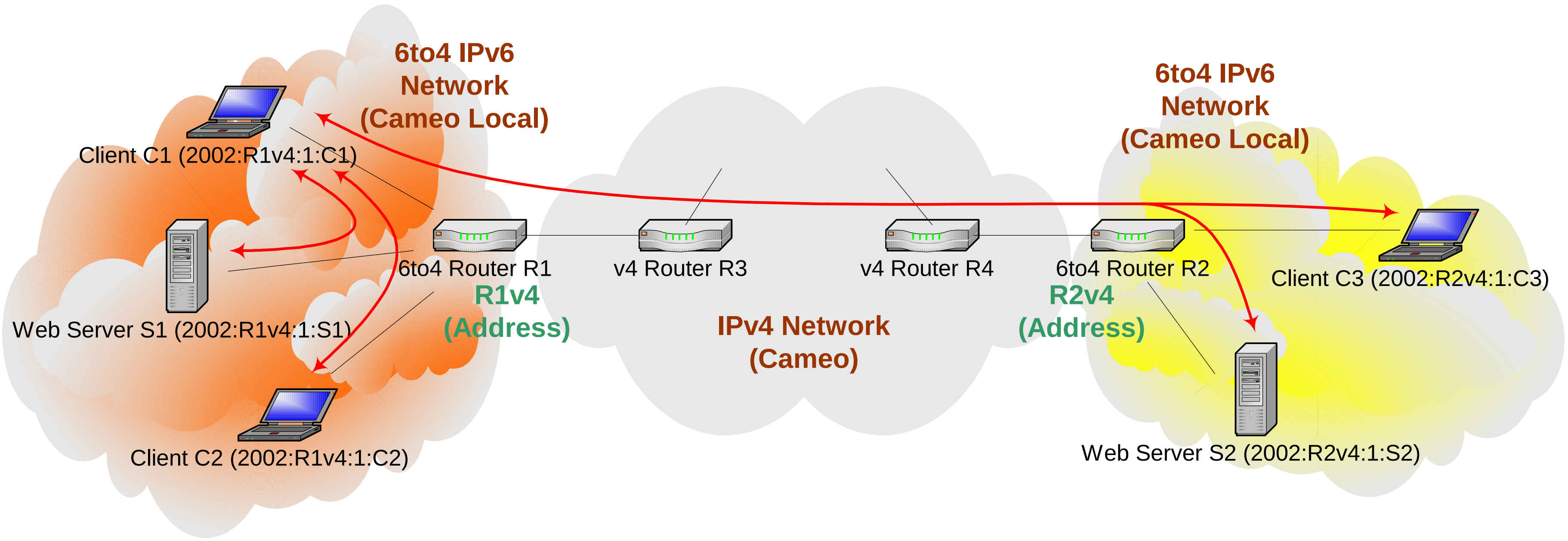
Tunneling Mechanism (8)
(6to4 Tunnel)

- Client user automatically creates a 6to4 IPv6 address using 6to4 router advertisement message.



Tunneling Mechanism (9) (6to4 Tunnel)

- 6to4 tunnel mechanism test environment (in Cameo company).
- Each node gets 6to4 ipv6 address using stateless autoconfiguration.



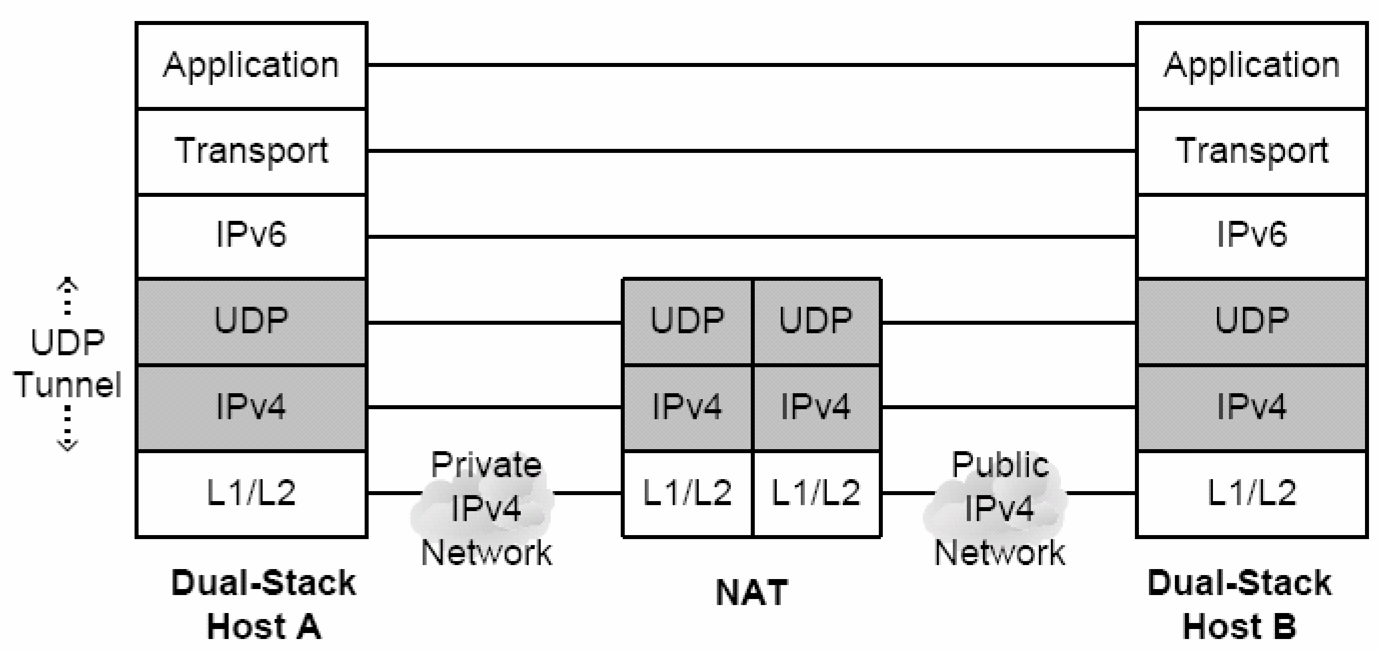
Tunneling Mechanism (10)
IPv6 6to4 Ping Test

- (ICMP test) in Windows XP:
- From 2002:ac15:2179:1:1c46:fb8b:b3f0:f59c to 2002:ac15:221f:1:755a:7783:8a01:765b.

```
D:\WINDOWS\system32\cmd.exe

D:\Documents and Settings\norp>ping6 2002:ac15:221f:1:755a:7783:8a01:765b

Pinging 2002:ac15:221f:1:755a:7783:8a01:765b
from 2002:ac15:2179:1:1c46:fb8b:b3f0:f59c with 32 bytes of data:

Reply from 2002:ac15:221f:1:755a:7783:8a01:765b: bytes=32 time=1ms
Reply from 2002:ac15:221f:1:755a:7783:8a01:765b: bytes=32 time<1ms
Reply from 2002:ac15:221f:1:755a:7783:8a01:765b: bytes=32 time<1ms
Reply from 2002:ac15:221f:1:755a:7783:8a01:765b: bytes=32 time<1ms

Ping statistics for 2002:ac15:221f:1:755a:7783:8a01:765b:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms

D:\Documents and Settings\norp>
```

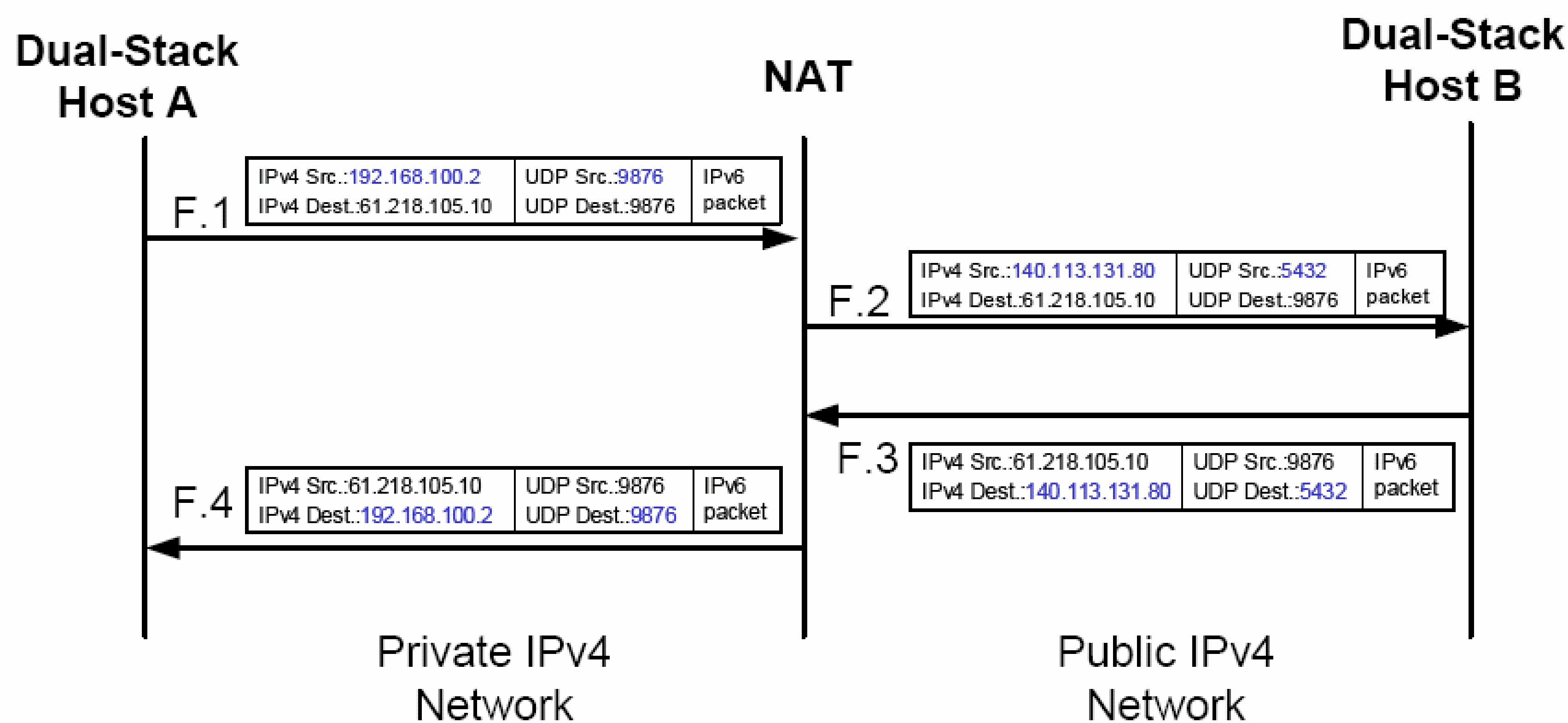
Tunneling Mechanism (11)
Teredo Tunnel

- UDP tunnel:
  - Protocol stack of UDP tunnel with NAT.



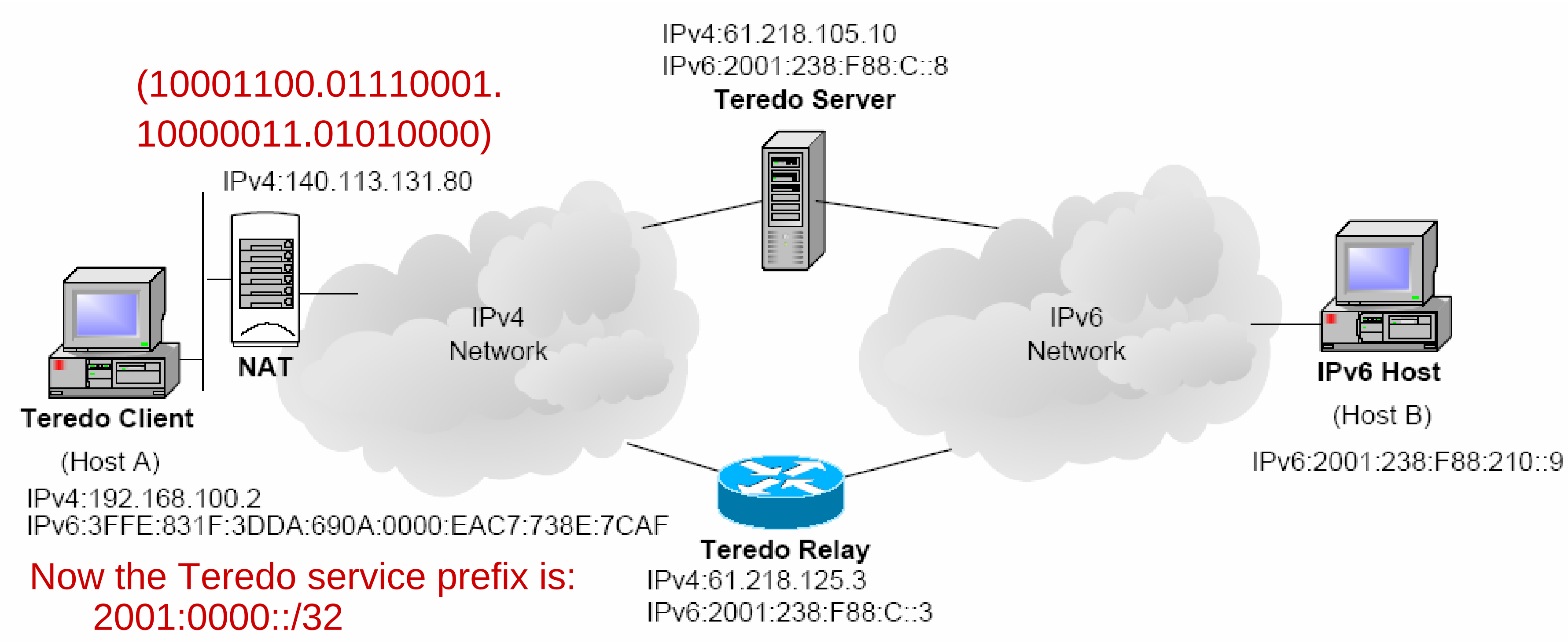
Tunneling Mechanism (12)
Teredo Tunnel

- Packet transmission using UDP tunnel.



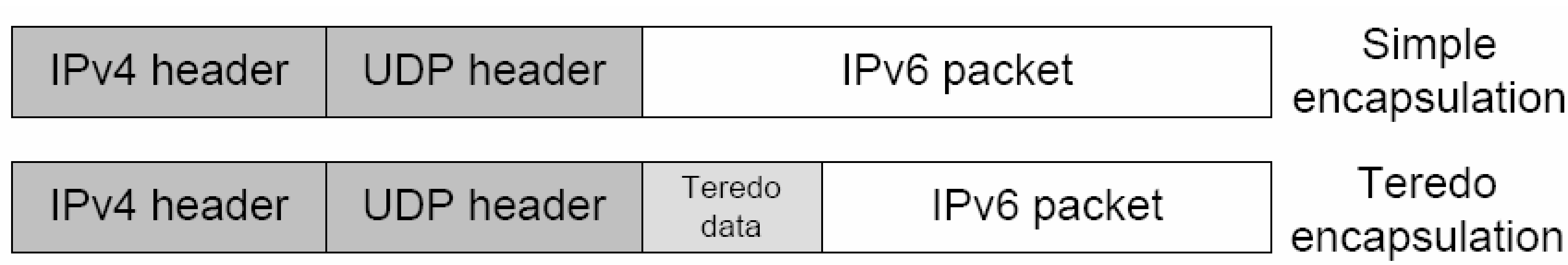
Tunneling Mechanism (13) Teredo Tunnel

- Teredo tunnel architecture.



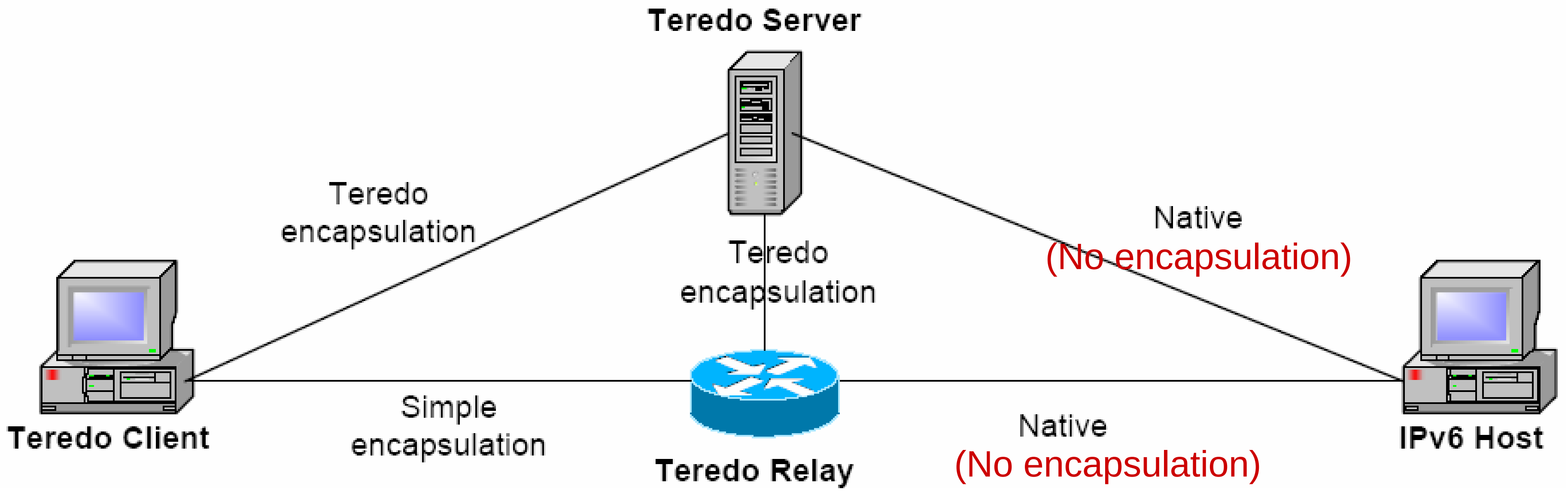
Tunneling Mechanism (14)
Teredo Tunnel

- Teredo encapsulation packet formats.



Tunneling Mechanism (15) Teredo Tunnel

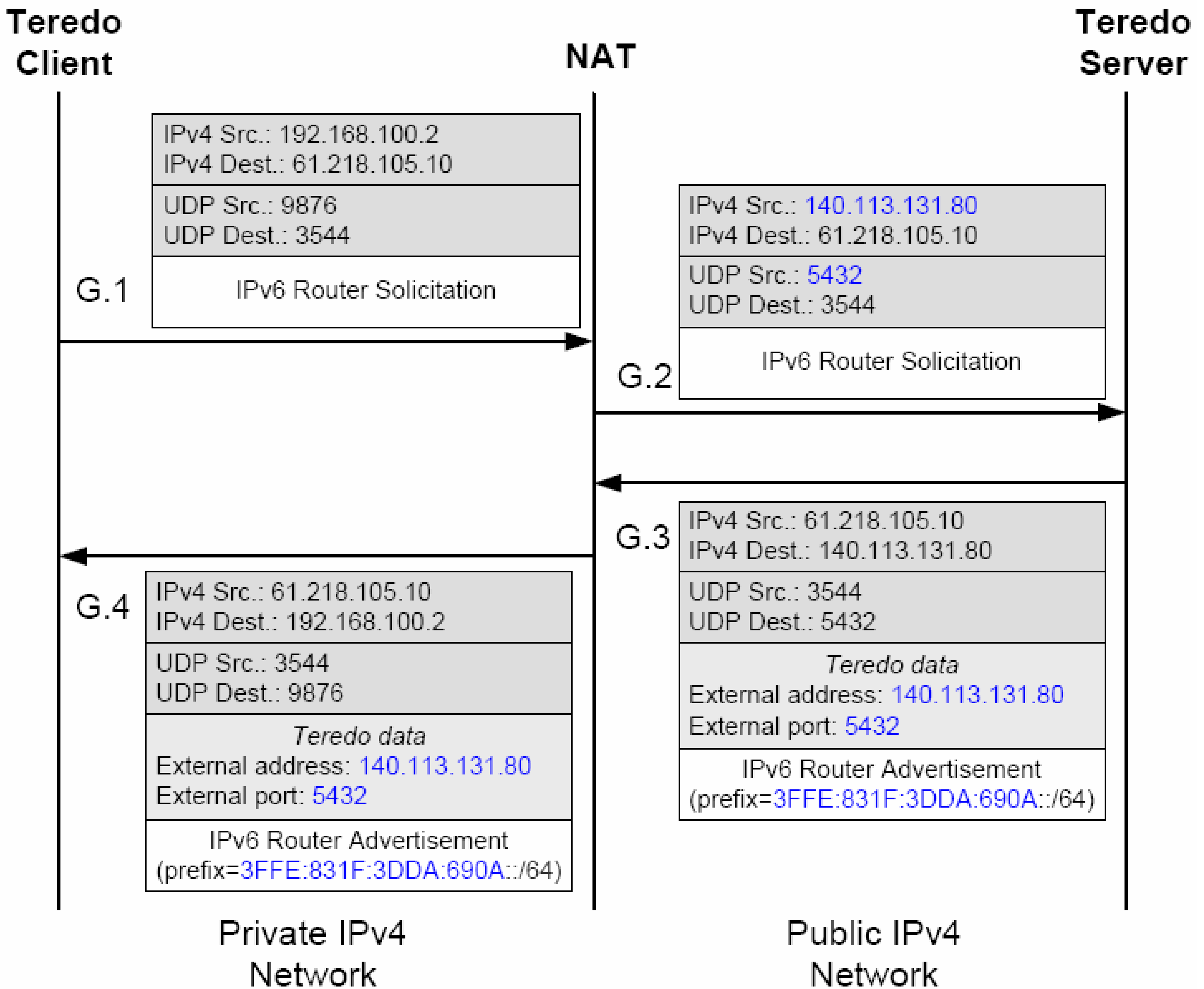
- Encapsulation formats used between Teredo nodes.



Tunneling Mechanism (16)
Teredo Tunnel

- On booting: Teredo client qualification procedure:

  Teredo client IPv6 address acquiring.
- Using RS & RA message.



Tunneling Mechanism (17) Teredo Tunnel

- Packet transmission is from host B to host A.

Step H.1. B 送 IPv6 封包給 A，IPv6 封包的來源位址及目的位址分別為 2001:238:F88:210::9 及 3FFE:831F:3DDA:690A:0000:EAC7:738E:7CAF。此封包透過 IPv6 routing protocol 被送往離 B 最近的 Teredo Relay。

Step H.2. Teredo Relay 收到此 IPv6 封包。若 Teredo Relay 會轉送 IPv6 封包給 A，則 Teredo Relay 可以直接用 Simple encapsulation 轉送 IPv6 封包給 A (跳至 **Step H.7**)。若 Teredo Relay 不會轉送 IPv6 封包給 A，基於 port-restricted NAT 的特性，必須要 Teredo Client 會透過 NAT 與 Teredo Relay 通訊後，Teredo Relay 才能藉由 NAT 送資料給 Teredo Client。因此，Teredo Relay 先把要送給 Teredo Client 的 IPv6 封包暫存至 buffer，並向 Teredo Server 送出以 Teredo encapsulation 封裝的 Teredo Bubble。Teredo Bubble 的來源 IPv6 位址和目的 IPv6 位址分別為

Tunneling Mechanism (18)
Teredo Tunnel

2001:238:F88:210::9 及 3FFE:831F:3DDA:690A:0000:EAC7:738E:7CAF。Teredo Bubble 內 Teredo data 夾帶的來源 IPv4 位址和來源 port 爲 Teredo Relay 的 IPv4 位址(61.218.125.3)和 port 3544。Teredo encapsulation 封裝的 UDP 來源 port 和目的 port 皆設爲 3544，IPv4 來源位址設爲 61.218.125.3，目的位址設爲 Teredo Server 的 IPv4 位址(可由目的 IPv6 位址中的 Teredo Server IPv4 位址欄位換算得知：0x3DDA690A=61.218.105.10)。

Step H.3. Teredo Server 從 port 3544 收到 Teredo Relay 送過來的 Teredo Bubble，TeredoServer 以 Teredo encapsulation 轉送此 Teredo Bubble 給 Teredo Client。Teredo Server 將自己的 IPv4 位址(61.218.105.10)及 port 3544 設爲 Teredo encapsulation 的來源 IPv4 位址及來源 port。目的 IPv4 位址和目的 port 則由目的 IPv6 位址計算得出(Obfuscated Teredo Client IPv4 位址欄位爲 0x738E7CAF，還

Tunneling Mechanism (19)
Teredo Tunnel

原後得到 0x8C718350=140.113.131.80；Obfuscated Teredo Client Port 欄位爲 0xEAC7，還原後得到 0x1538=5432)。

Step H.4. NAT 收下內含 Teredo Bubble 的 UDP 封包。NAT 根據表 3-1 轉換封包欄位內容。

Step H.5. Teredo Client 收到 Teredo Relay 送來的 Teredo Bubble 後，用 Simple encapsulation 回送 Teredo Bubble 給 Teredo Relay。此 Teredo Bubble 的來源位址和目的位址分別爲 3FFE:831F:3DDA:690A:0000:EAC7:738E:7CAF 和 2001:238:F88:210::9，Simple encapsulation 的來源 IPv4 位址和來源 port 設爲 192.168.100.2 和 9876，目的 IPv4 位址和目的 port 則設爲 Teredo data 中紀錄的來源 IPv4 位址 (61.218.125.3)和來源 port (3544)值。

Tunneling Mechanism (20)
Teredo Tunnel

Step H.6. NAT 攔截內含 Teredo Bubble 的 Simple encapsulation 封包。NAT 根據此封包建立新的位址對應表(如表 3-2)，並在根據此對應表轉換封包欄位內容後，將此封包送給 Teredo Relay。

Step H.7.此時 NAT 已建立位址對應表。Teredo Relay 將 IPv6 封包用 Simple encapsulation 轉送 IPv6 封包至 Teredo Client(若有送向 Teredo Client 的 IPv6 封包被 Teredo Relay 暫存在 buffer，也在此時將 buffer 中的 IPv6 封包一併傳送)。Simple encapsulation 封包的來源 IPv4 位址和來源 port 為 Teredo Relay 的 IPv4 位址(61.218.125.3)和 3544，目的 IPv4 位址和目的 port 則由目的 IPv6 位址(Teredo Client 的 IPv6 位址)計算得到：還原目的 IPv6 位址的 Obfuscated Teredo Client IPv4 位址欄位得到目的 IPv4 位址為 140.113.131.80 (0x738E7CAF ⊕ 0xFFFFFFFF=0x8C718350=140.113.131.80)，還原目的 IPv6 位址的 Obfuscated Teredo Client Port 欄位得到目的 port 為 5432 (0xEAC7 ⊕ 0xFFFF=0x1538=5432)。

Tunneling Mechanism (21)
Teredo Tunnel

Step H.8. NAT 收下內含 IPv6 封包的 UDP 封包，NAT 根據表 3-2 轉換封包欄位內容。

Step H.9. Teredo Client 收到從 B 送來，由 Teredo Relay 轉送的 Simple encapsulation 封包。

Teredo Client 紀錄離 B 最近的 Teredo Relay 爲 61.218.125.3。往後當 A 要送 IPv6 封包給 B 時，Teredo Client 由此可知離 B 最近的 Teredo Relay 爲 61.218.125.3。

Teredo Client 以 Simple encapsulation 傳送 IPv6 封包(設定來源 IPv4 位址和來源 port 分別爲 192.168.100.2 和 9876，目的位址和目的 port 分別爲 61.218.125.3 和 3544)。

Step H.10. NAT 收下內含 IPv6 封包的 Simple encapsulation 封包，NAT 根據表 3-2 轉換封包欄位內容。

Step H.11. Teredo Relay 收到此封包，將 Simple encapsulation 內的 IPv6 封包送至 IPv6 網路。此封包被 B 收下。

Tunneling Mechanism (22)
Teredo Tunnel

接著，A 和 B 之間就可以用 **Steps H.1, H.7-H.11** 的步驟互相傳送 IPv6 封包。

若 Teredo Relay 收到目的 IPv6 位址 Flags 值爲 0x8000 的 IPv6 封包(Flags 爲 0x8000 表示 NAT 上位址對應表爲表 3-3 這種型式，不限制任何 Remote 欄位的值)，Teredo Relay 可以直接用 **Steps H.1, H.7-H.11** 的流程將 IPv6 封包用 Simple encapsulation 轉送給 Teredo Client，不需爲了建立 Teredo Relay 的位址對應表傳送 Teredo Bubble，**Steps H.2-H.6** 這幾個步驟可以省略。

Tunneling Mechanism (23) Teredo Tunnel

- Packet transmission is from host A to host B.

Step I.1. A 把 IPv6 封包傳遞給 Teredo Client，由 Teredo Client 負責將封包送至 B。由於 B 不會送 IPv6 封包給 Teredo Client，Teredo Client 不知道離 B 最近 Teredo Relay 的 IPv4 位址。A 送給 B 的 IPv6 封包會被存放至 Teredo Client 的 buffer。Teredo Client 用 Simple encapsulation 向 Teredo Server 送出目的 IPv6 位址為 B、來源 IPv6 位址為 A 的 ICMPv6 Echo Request。

Step I.2. 內含 ICMPv6 Echo Request 的 Simple encapsulation 封包被 NAT 攔截，NAT 根據表 3-1 將此封包作位址轉換後，將此封包送至 Teredo Server。

Step I.3. Teredo Server 從 port 3544 收到此封包，將 Simple encapsulation 裡的 ICMPv6 Echo Request 送至 IPv6 網路。

Tunneling Mechanism (24)
Teredo Tunnel

Step I.4. B 收到 Teredo Server 轉送過來的 ICMPv6 Echo Request，回送 ICMPv6 Echo Response 給 A。此 IPv6 封包透過 IPv6 routing protocol，找到離 B 最近的 Teredo Relay。(若此 Teredo Relay 不曾送 IPv6 封包給 Teredo Client，Teredo Relay 會執行傳送 Teredo Bubble 的流程，請參照 **Steps H.2-H.6**)

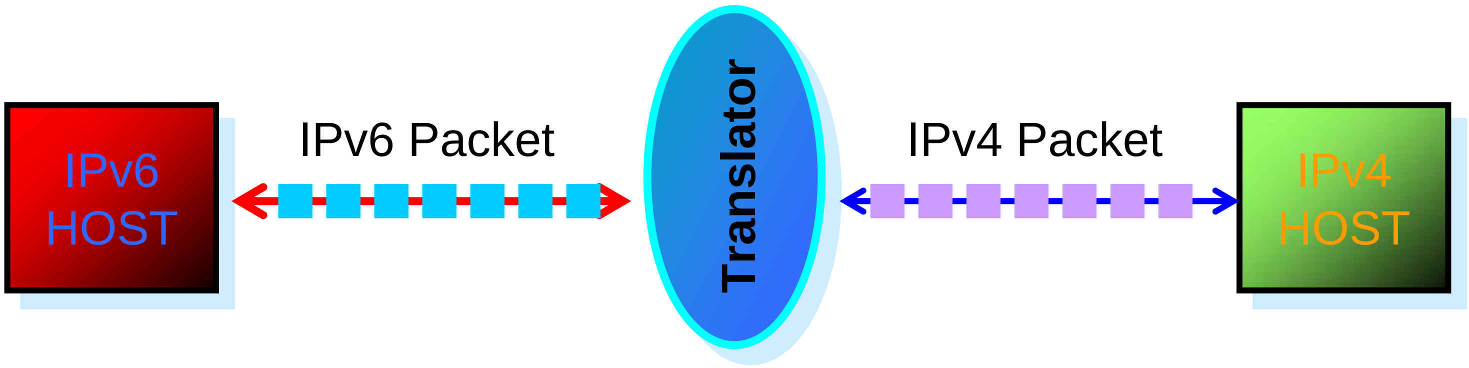
Step I.5. Teredo Relay 以 Simple encapsulation 方式將 ICMPv6 Echo Response 傳送給 Teredo Client。封包的目的 IPv4 位址和目的 port 設定和 **Step H.7** 相同。此 UDP 封包被送往 NAT。

Step I.6. 內含 ICMPv6 Echo Response 的 Simple encapsulation 封包被 NAT 攔截，NAT 根據表 3-2 將此封包作位址轉換後，將此封包送至 Teredo Client。Teredo Client 收到來自 B 的 ICMPv6 Echo Response，Teredo Client 紀錄離 B 最近的 Teredo Relay 爲轉送此 ICMPv6 Echo Response 的 Teredo Relay (61.218.125.3)。

接著，A 和 B 之間就可以用 **Steps H.1, H.7-H.11** 的流程互相傳送 IPv6 封包。

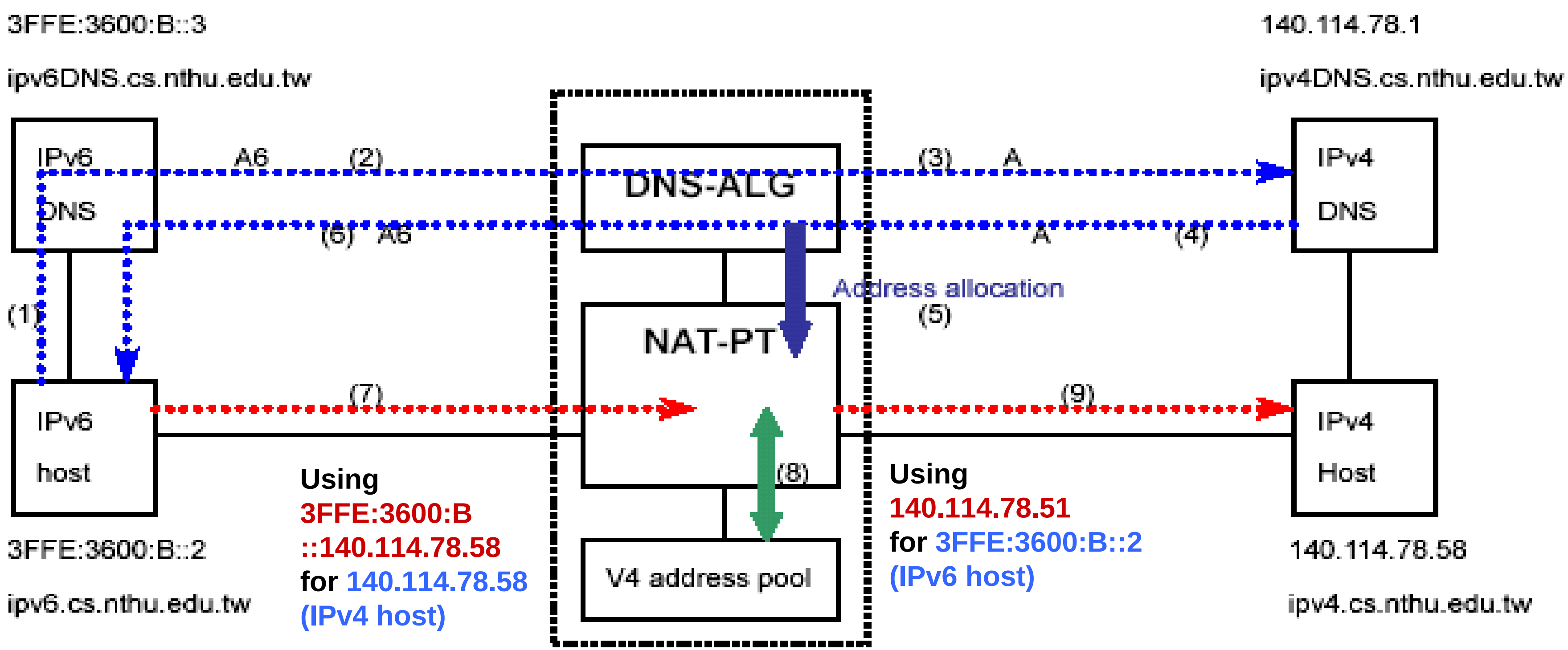
Translator Mechanism (1)

- IPv4 packet and IPv6 packet is mutually translated
  - Translated at IP layer (NAT-PT/NAPT-PT, SIIT).
  - Translated at transport layer (TCP-UDP relay mechanism).
  - Translated at application layer (BIS/BIA, Socks).
  - Etc.



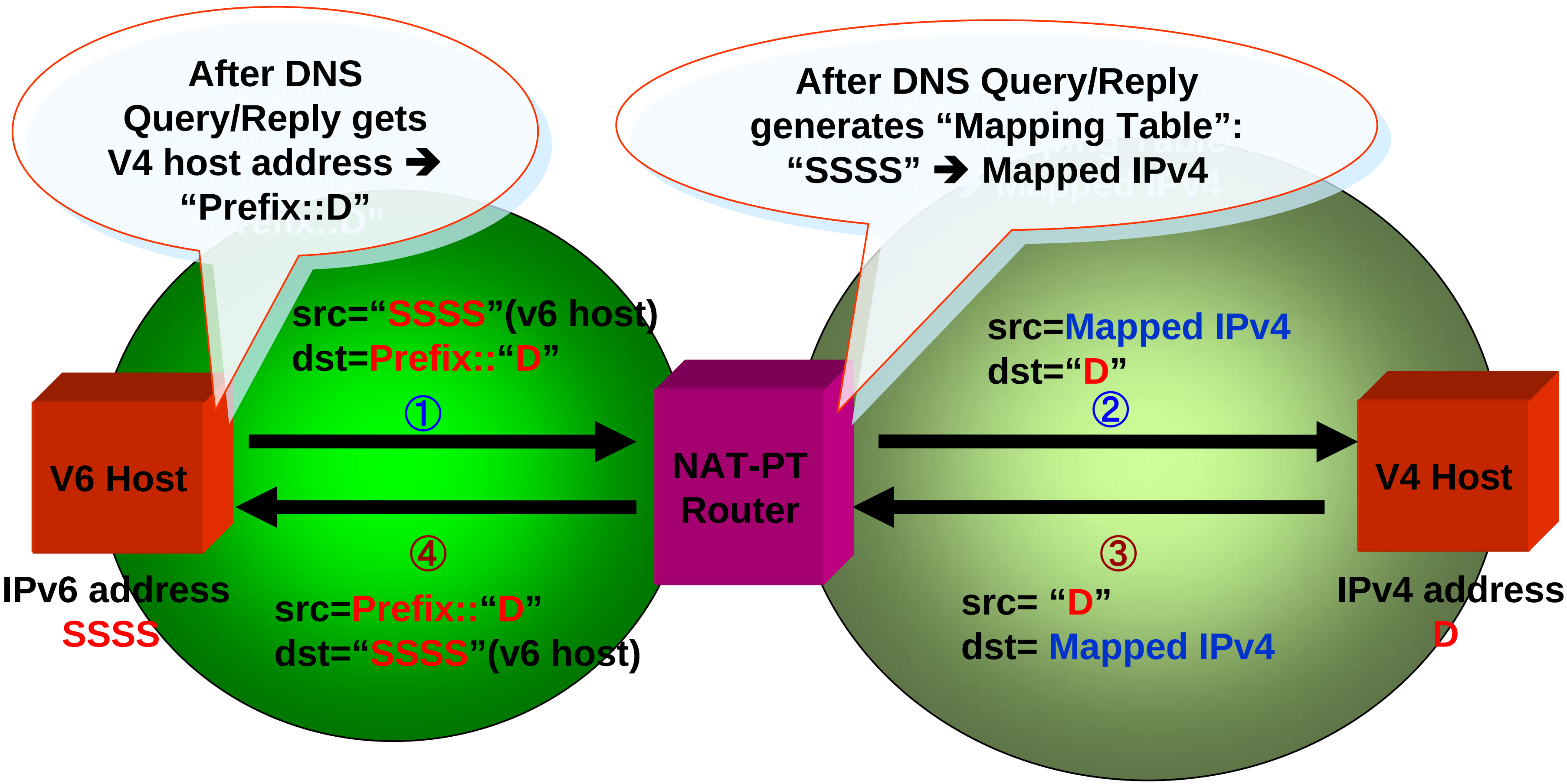
Translator Mechanism (2) (NAT-PT/NAPT-PT)

- DNS-ALG ALG (Application Level Gateway):
  - help NAT-PT to establish automated IP mapping function.
- Exchange DNS message: type A (for v4) ←→ type AAAA/A6 (for v6).
  - DNS message (Name, Value, Type, TTL).
  - IF type=A represents: name=URL address & value=IPv4 address.



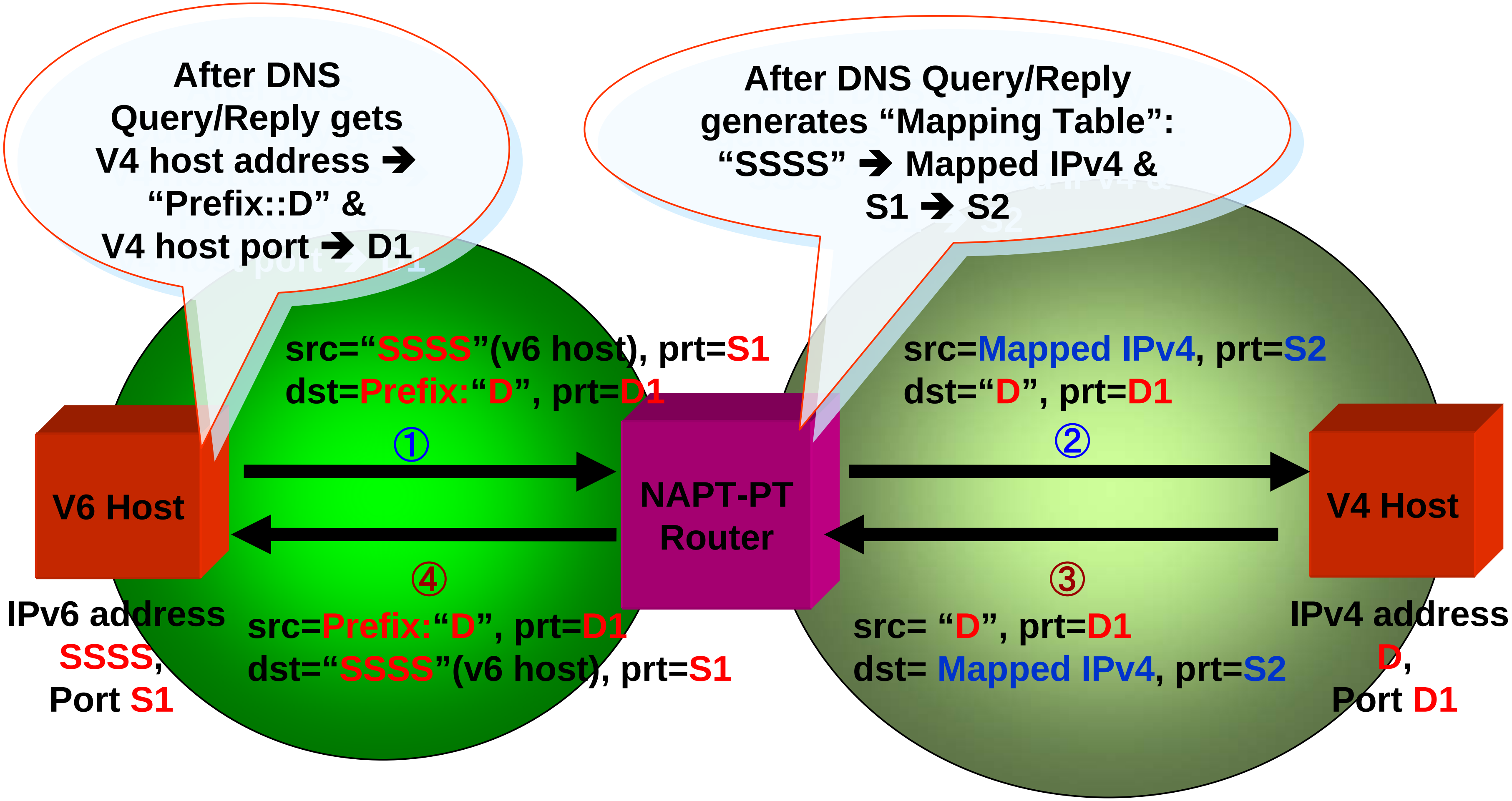
Translator Mechanism (3) (NAT-PT/NAPT-PT)

- NAT-PT operation (v6 host connects to v4 host).
- EX) After DNS query/reply operation.



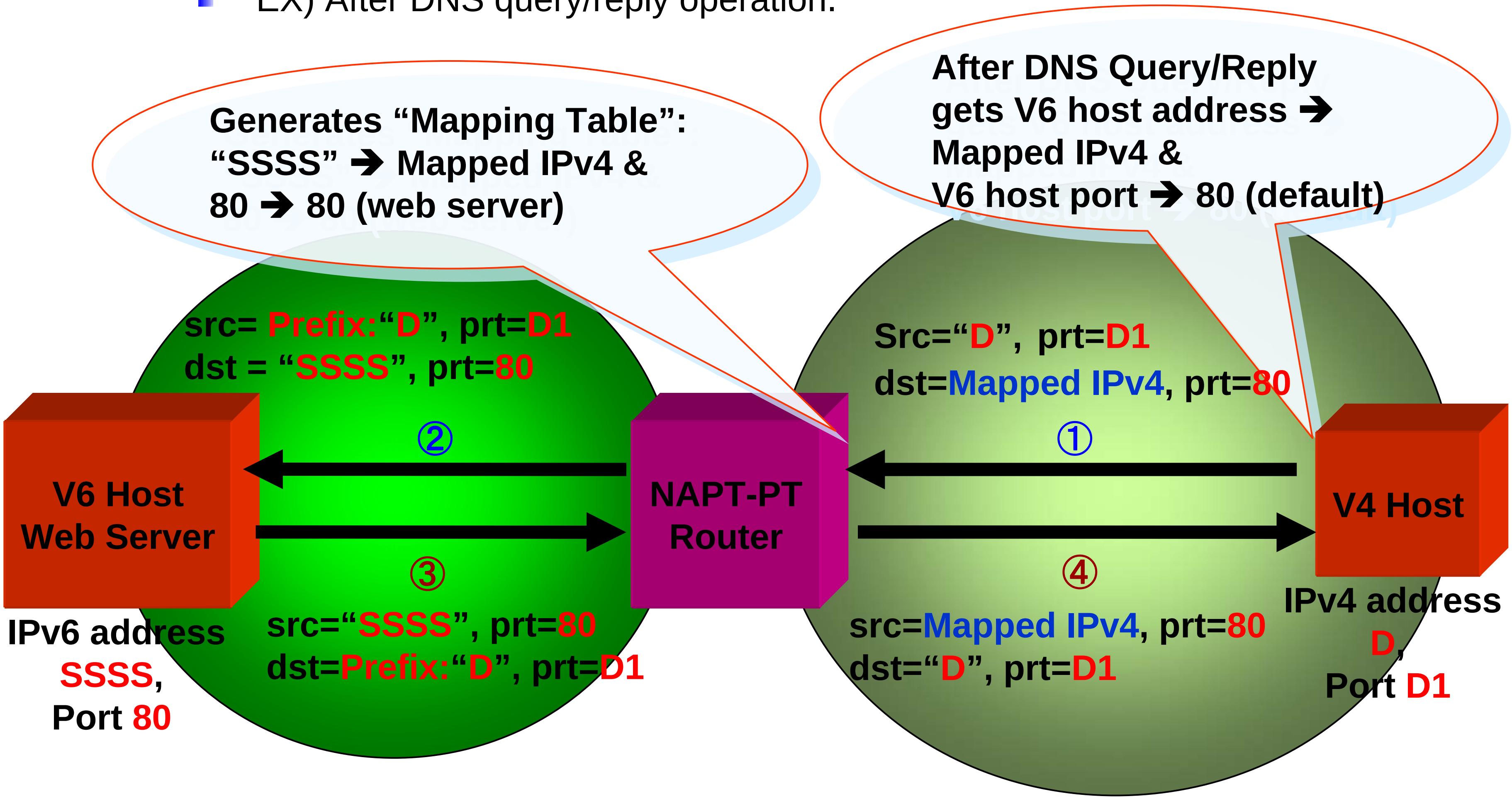
Translator Mechanism (4) (NAT-PT/NAPT-PT)

- NAPT-PT operation (v6 host connects to v4 host).
- EX) After DNS query/reply operation.



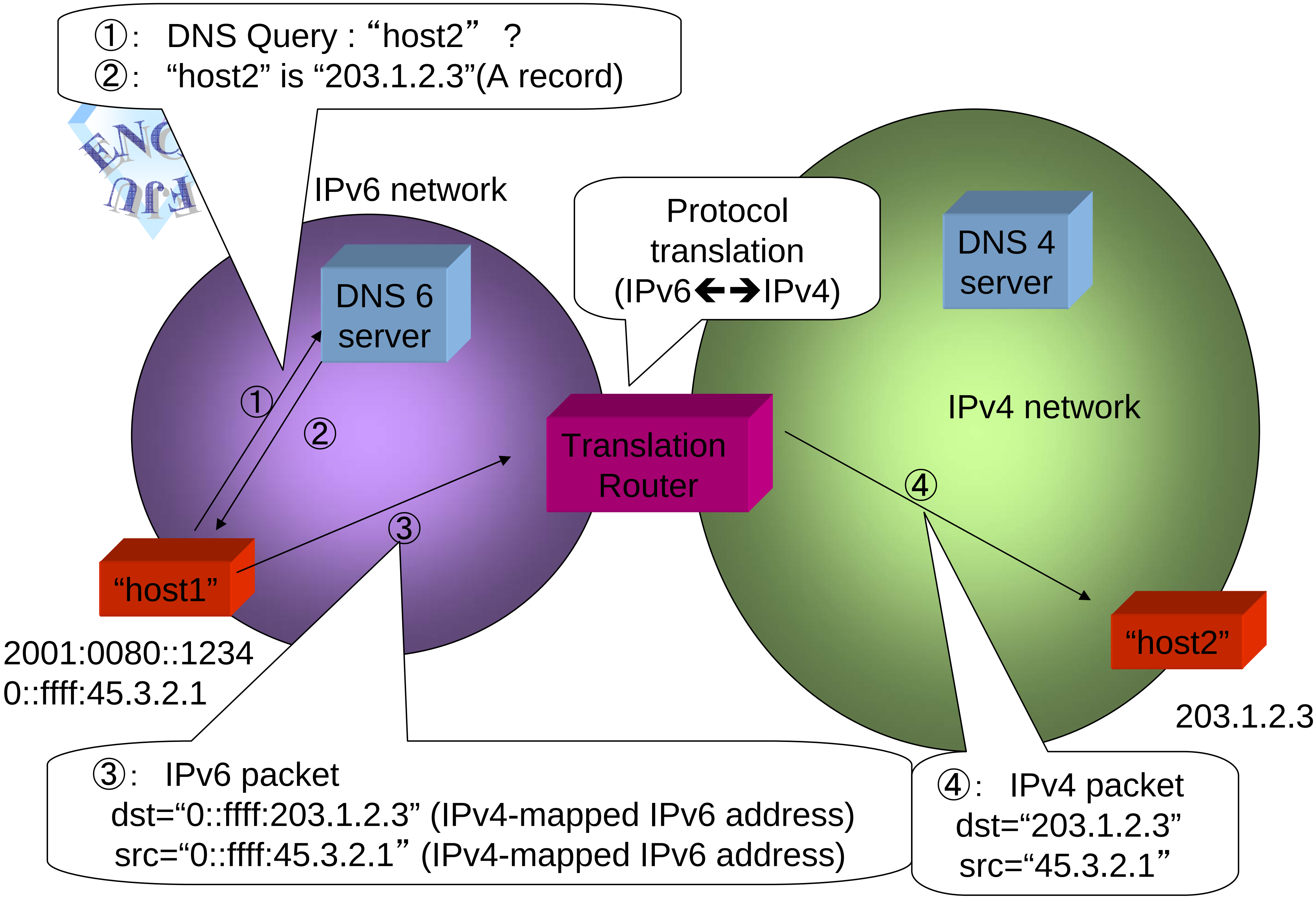
Translator Mechanism (5) (NAT-PT/NAPT-PT)

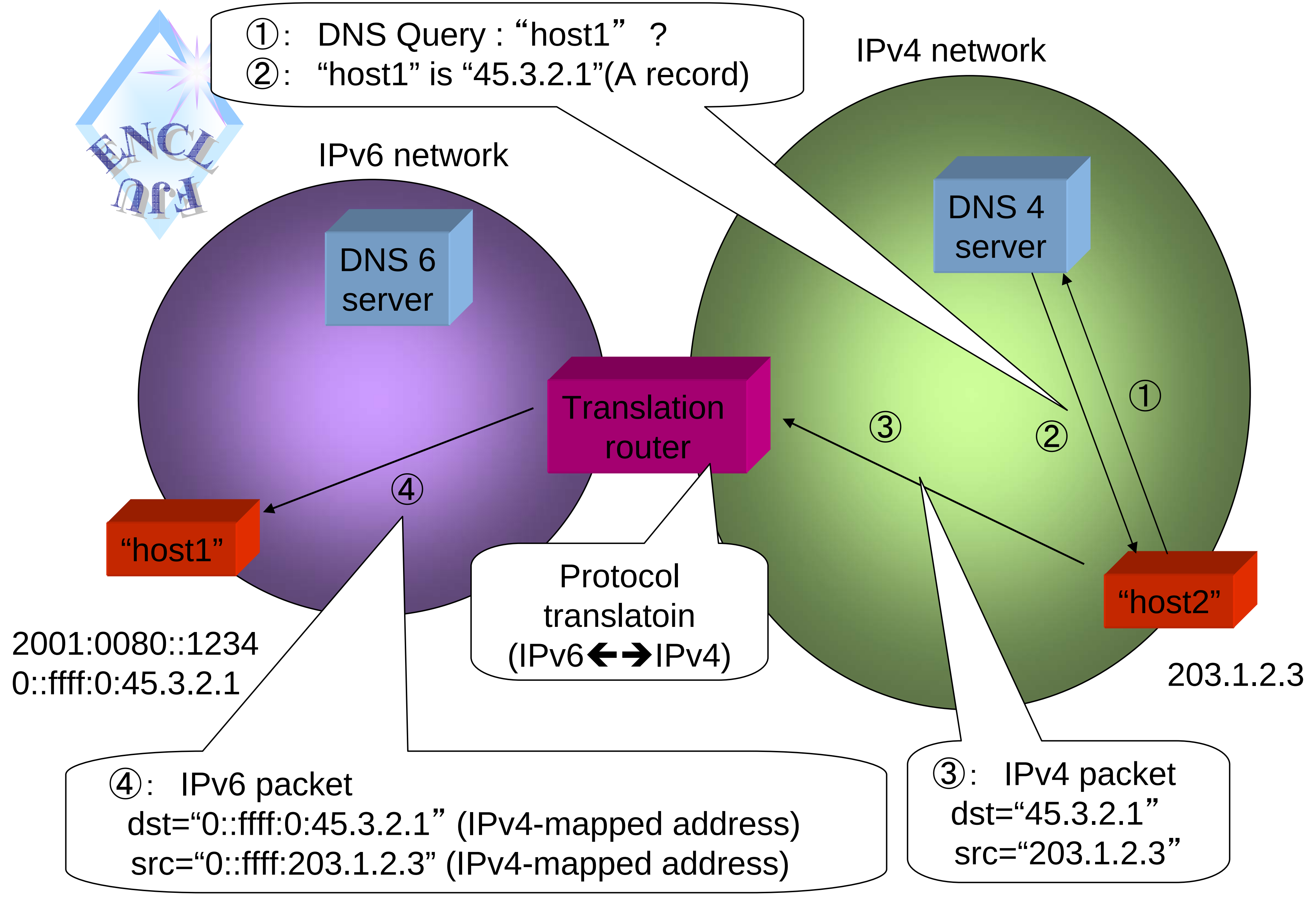
- NAPT-PT operation (v4 host connects to v6 host, a web server, port=80).
- EX) After DNS query/reply operation.



Translator Mechanism (6)
SIIT (Stateless IP/ICMP Translation)

- Allows IPv6-only hosts to talk to IPv4 hosts.
  - The IPv6 host has IPv4 embedded IPv6 address (ex: 2001:0080::12340::ffff:45.3.2.1).
- Translation on IP packet header (including ICMP headers) in separate translator boxes in the network without requiring any per-connection state in those boxes.
- For IPv4 host, using IPv4-mapped IPv6 address (::ffff:a.b.c.d).
- Most option fields can not be translated.
- Requires one temporary IPv4 address per host.
- Does not define IPv4 address allocation.

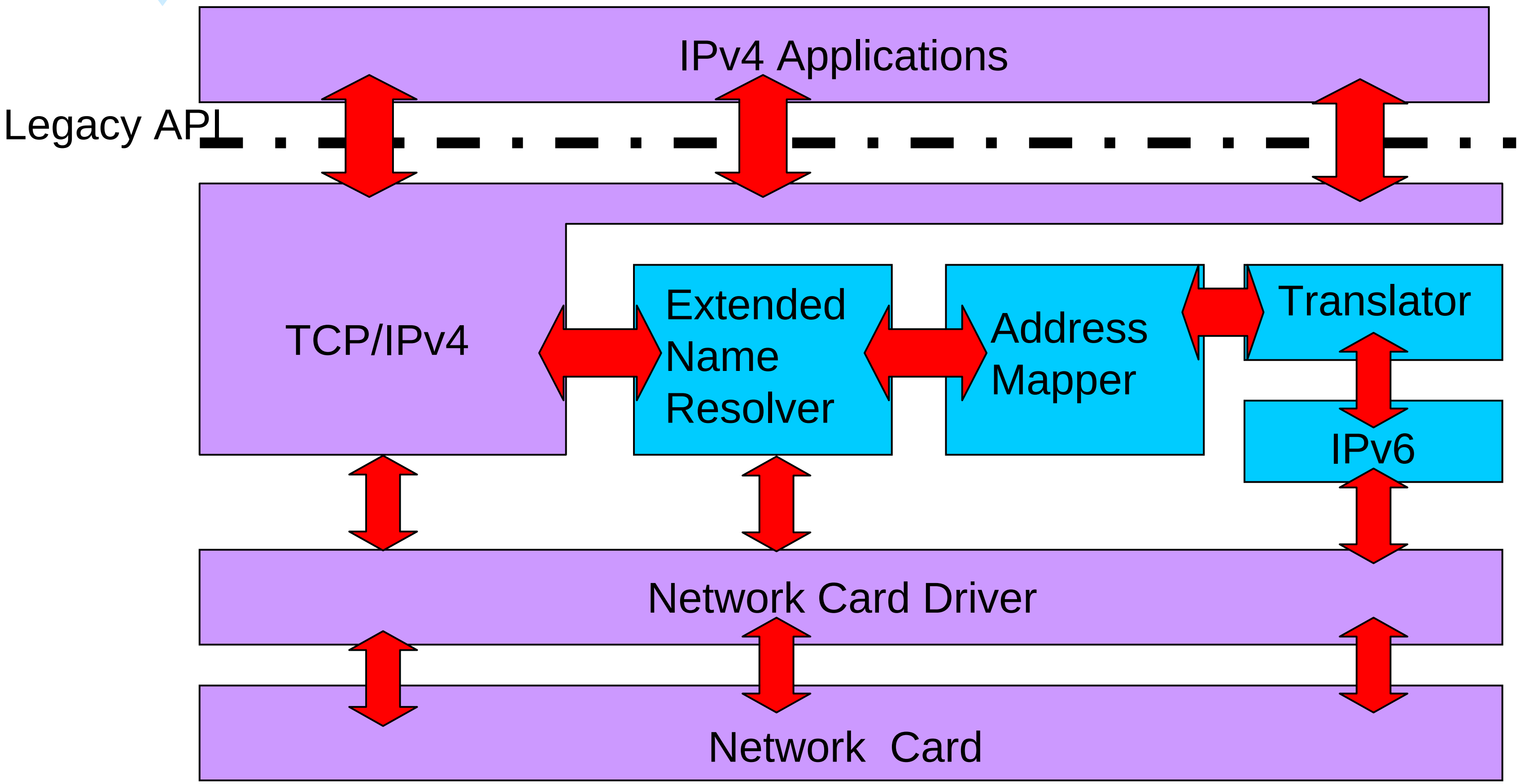

①: DNS Query : "host2" ?
②: "host2" is "203.1.2.3"(A record)
IPv6 network
DNS 6 server
Protocol translation (IPv6←→IPv4)
DNS 4 server
①
②
Translation Router
IPv4 network
④
③
"host1"
2001:0080::1234
0::ffff:45.3.2.1
"host2"
203.1.2.3
③: IPv6 packet
dst="0::ffff:203.1.2.3" (IPv4-mapped IPv6 address)
src="0::ffff:45.3.2.1" (IPv4-mapped IPv6 address)
④: IPv4 packet
dst="203.1.2.3"
src="45.3.2.1"


①: DNS Query : “host1” ?
②: “host1” is “45.3.2.1”(A record)
IPv4 network
IPv6 network
DNS 4 server
DNS 6 server
Translation router
①
②
③
④
“host1”
“host2”
Protocol translatoin (IPv6←→IPv4)
2001:0080::1234
0::ffff:0:45.3.2.1
203.1.2.3
④: IPv6 packet
dst=“0::ffff:0:45.3.2.1” (IPv4-mapped address)
src=“0::ffff:203.1.2.3” (IPv4-mapped address)
③: IPv4 packet
dst=“45.3.2.1”
src=“203.1.2.3”

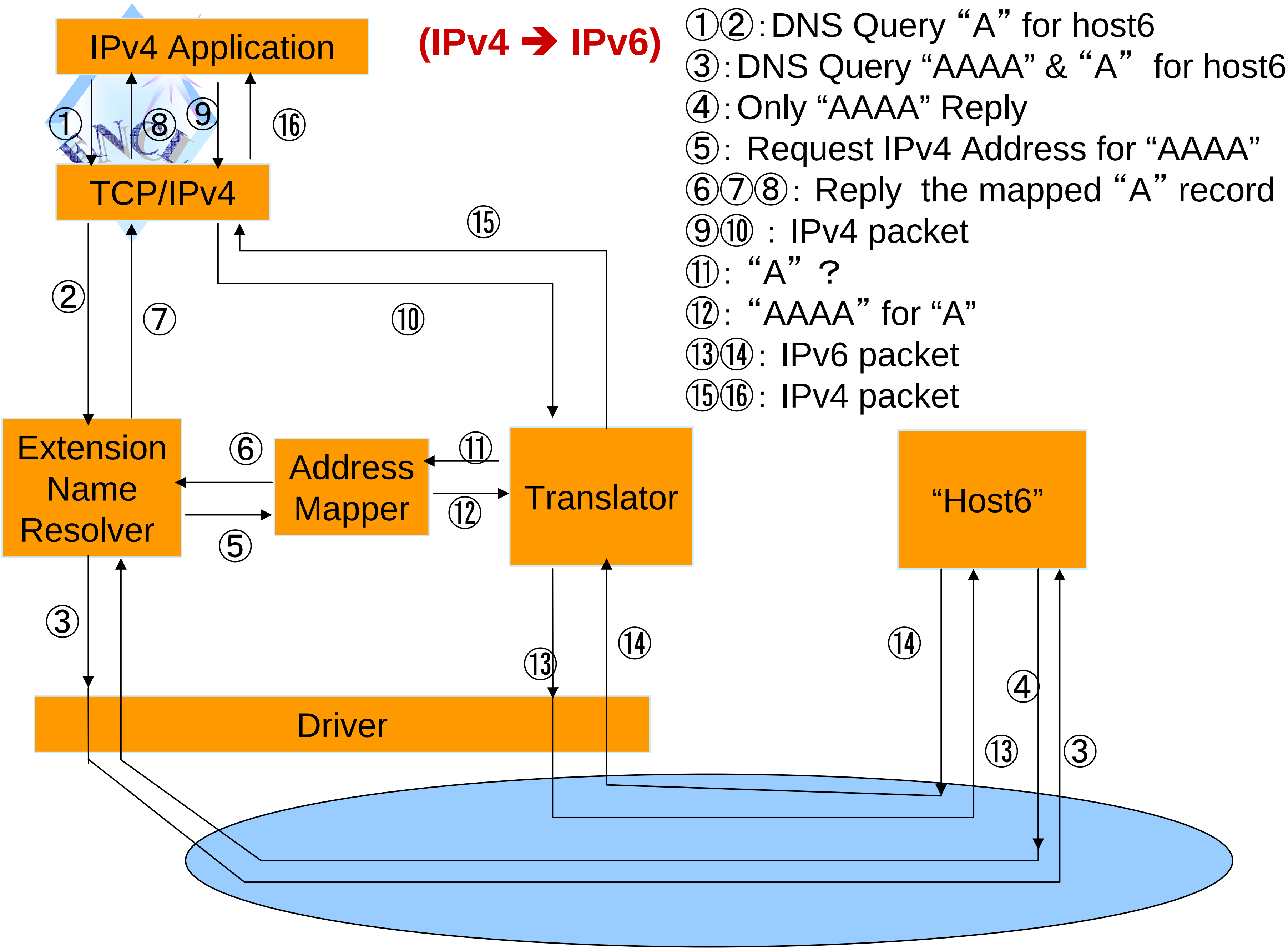
Translator Mechanism (7)
BIS (Bump-in-the-Stack)

- It is not a mechanism for packet transmission in networks, only focuses on applications translation in a host (Application layer).
- To trace data flow between TCP/IPv4 and network card driver.
- Early, many application programs do not have IPv6 function, or some users do not update application programs from IPv4 to IPv6 function.
- Dual stack architecture in a host.

Translator Mechanism (8)
BIS (Bump-in-the-Stack)

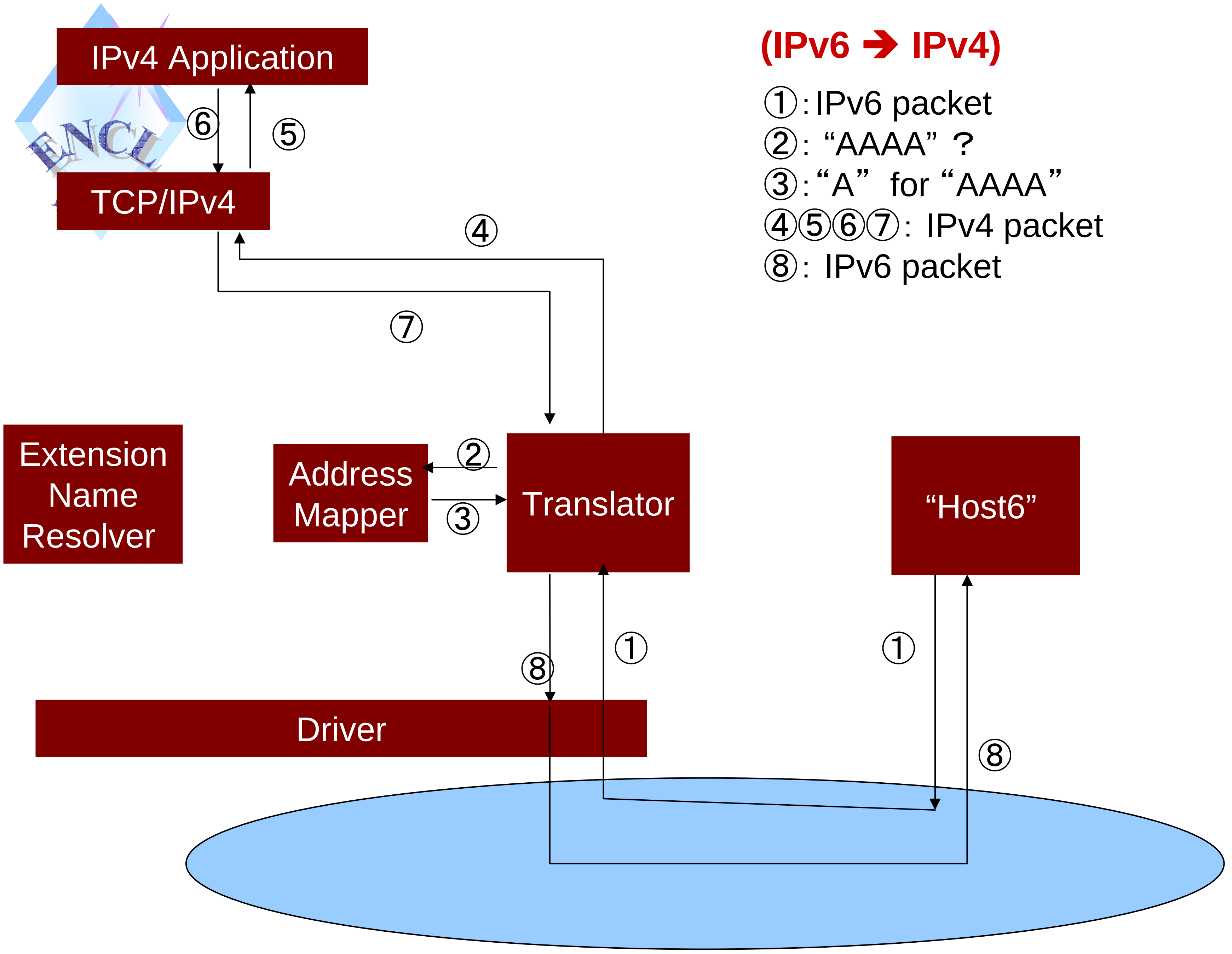


Bump in the stack (BIS) software structure


(IPv4 ➔ IPv6)
①②: DNS Query "A" for host6
③: DNS Query "AAAA" & "A" for host6
④: Only "AAAA" Reply
⑤: Request IPv4 Address for "AAAA"
⑥⑦⑧: Reply the mapped "A" record
⑨⑩ : IPv4 packet
⑪: "A" ?
⑫: "AAAA" for "A"
⑬⑭: IPv6 packet
⑮⑯: IPv4 packet
IPv4 Application
TCP/IPv4
Extension Name Resolver
Address Mapper
Translator
"Host6"
Driver
① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮ ⑯

(IPv6 → IPv4)

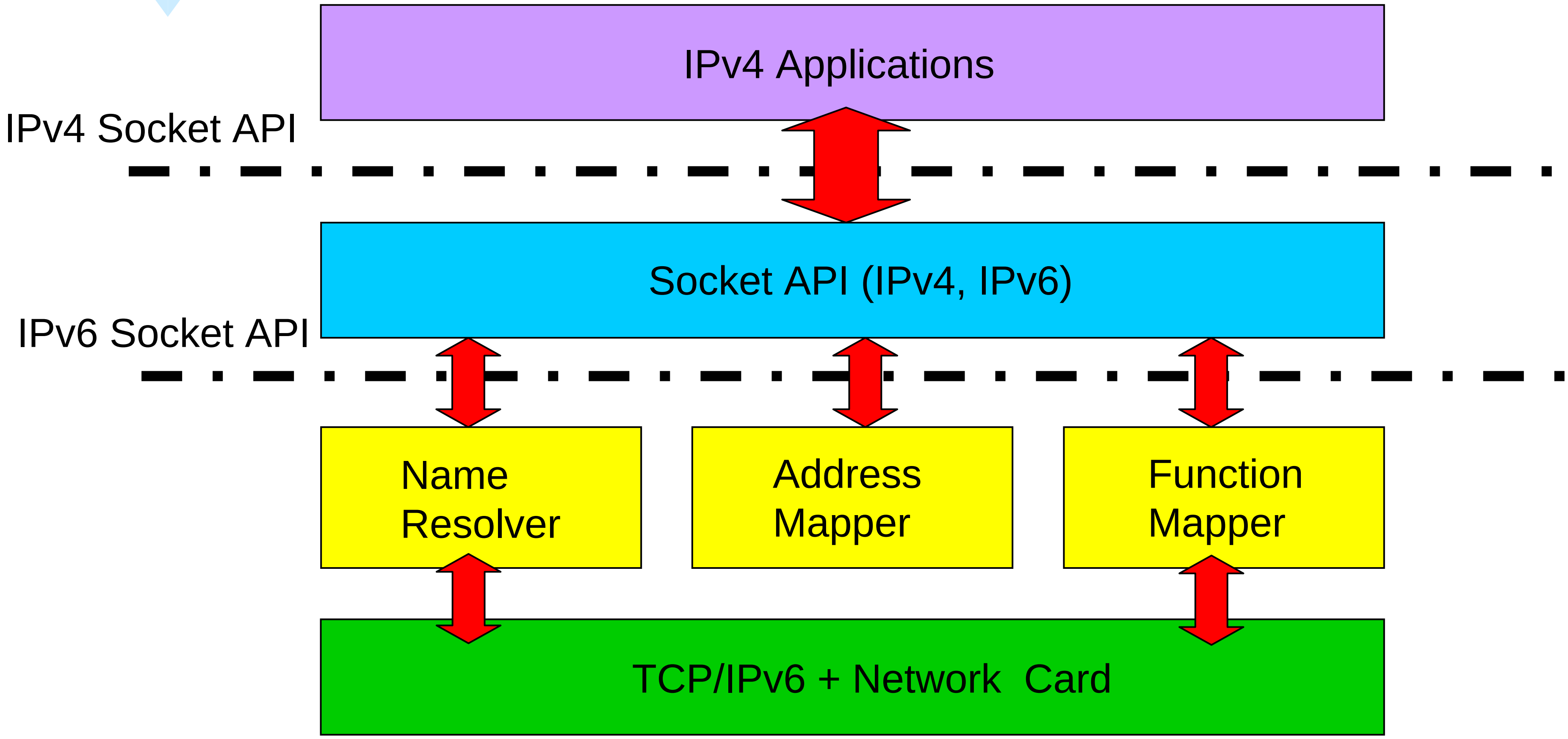
①: IPv6 packet
②: "AAAA" ?
③: "A" for "AAAA"
④⑤⑥⑦: IPv4 packet
⑧: IPv6 packet



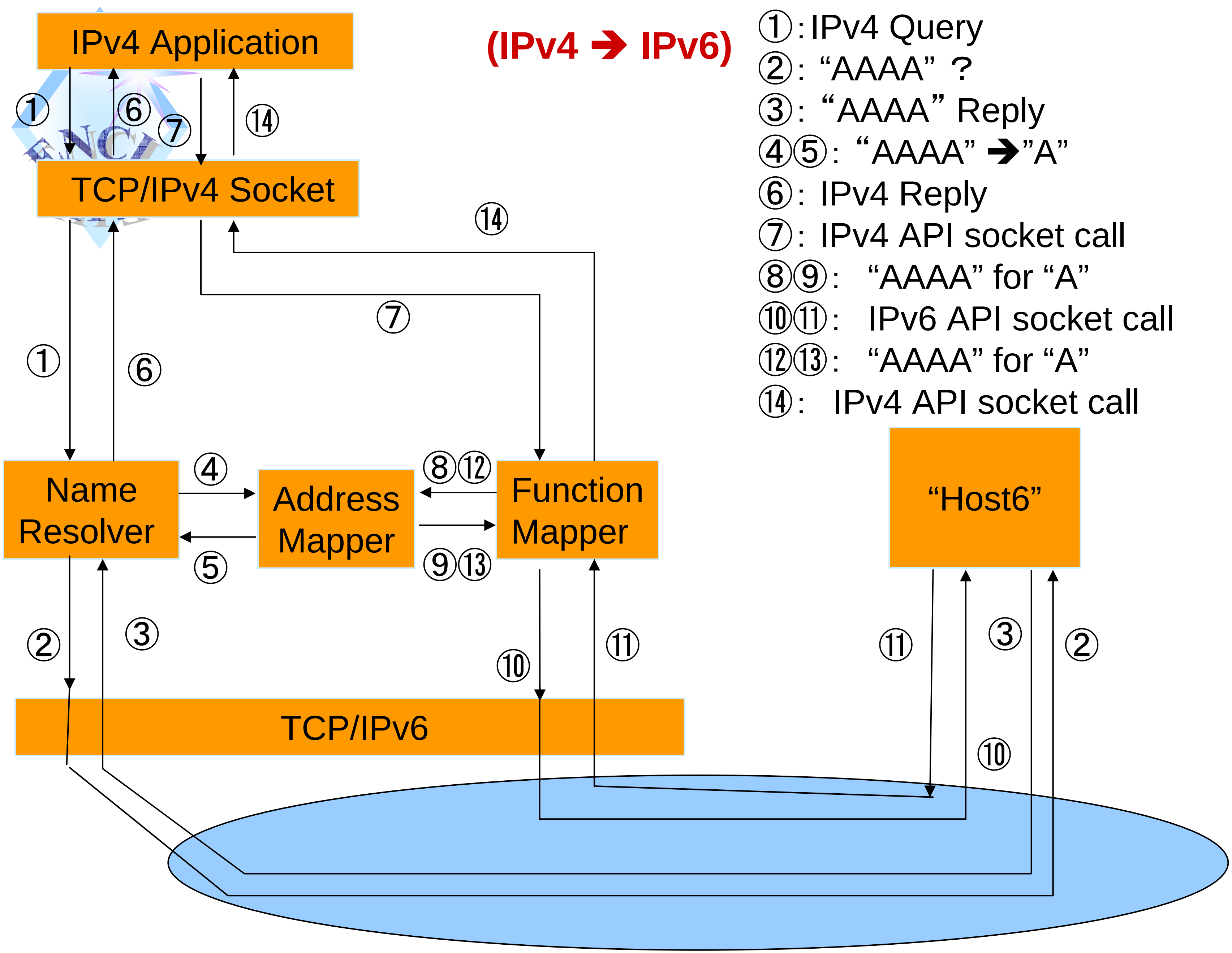
Translator Mechanism (9)
BIA (Bump-in-the-Application)

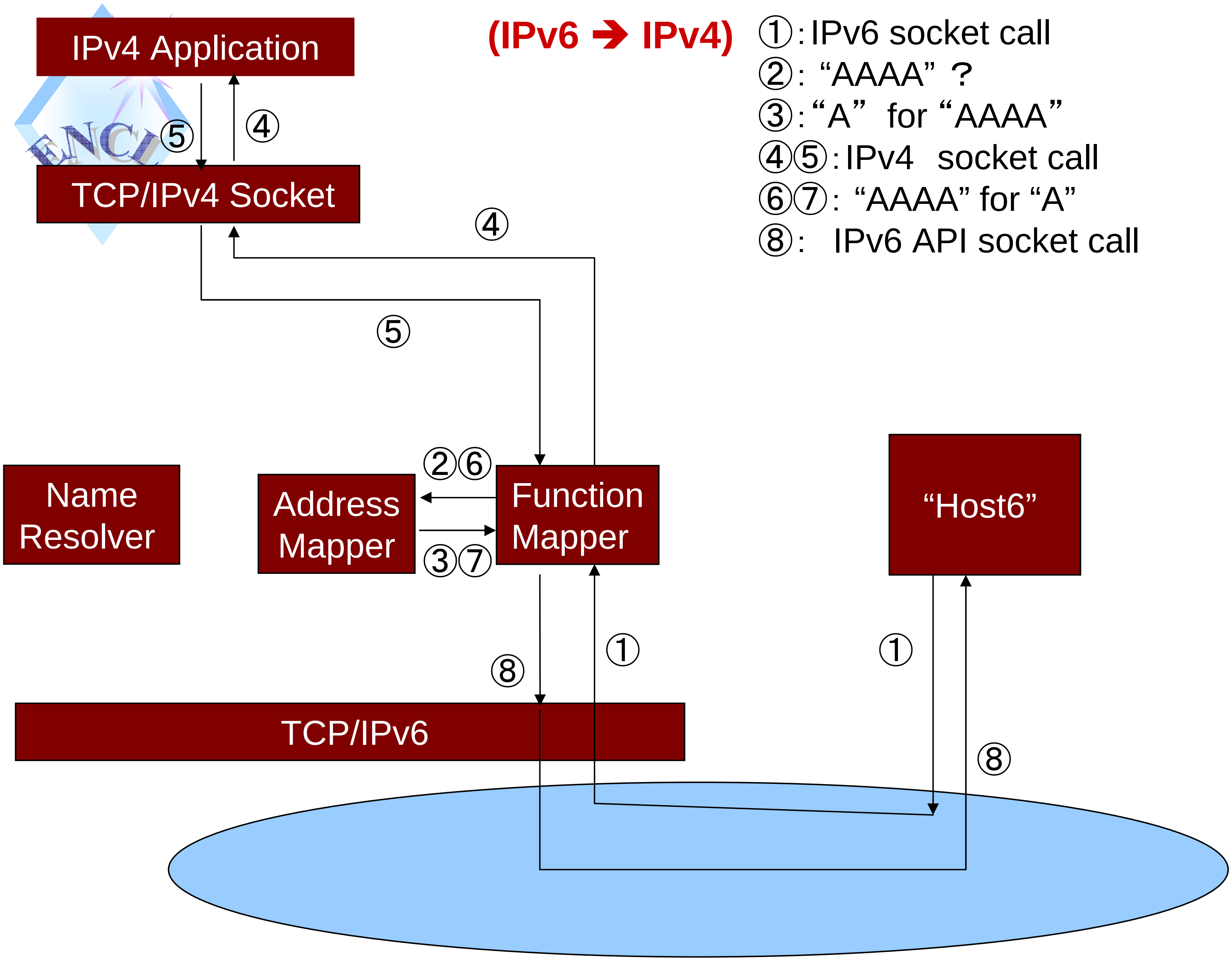
- Dual stack architecture in a host.
- Employ an API converter between TCP/IP and socket modules.

Translator Mechanism (10)
BIA (Bump-in-the-Application)



BIA (Bump in the Application) software structure


(IPv4 ➔ IPv6)
①: IPv4 Query
②: "AAAA" ?
③: "AAAA" Reply
④⑤: "AAAA" ➔"A"
⑥: IPv4 Reply
⑦: IPv4 API socket call
⑧⑨: "AAAA" for "A"
⑩⑪: IPv6 API socket call
⑫⑬: "AAAA" for "A"
⑭: IPv4 API socket call
IPv4 Application
TCP/IPv4 Socket
Name Resolver
Address Mapper
Function Mapper
"Host6"
TCP/IPv6
①
⑥
⑦
⑭
②
③
④
⑤
⑧⑫
⑨⑬
⑩
⑪


(IPv6 ➔ IPv4)
①: IPv6 socket call
②: "AAAA" ?
③: "A" for "AAAA"
④⑤: IPv4 socket call
⑥⑦: "AAAA" for "A"
⑧: IPv6 API socket call
IPv4 Application
TCP/IPv4 Socket
Name Resolver
Address Mapper
Function Mapper
"Host6"
TCP/IPv6
①
②⑥
③⑦
④
⑤
⑧

Translator Mechanism (11) (Using SOCKS)

- SOCKS server relays two "terminated" IPv4 and IPv6 connections at the "application layer".
  - Each socksified application has its own *Socks Lib*.
  - Replace applications' socket APIs and DNS name resolving APIs.