IPv6 Courses

©G6 Association

October 11, 2013

Table of Contents

Integration

Why IPv6 Integration ?

IPv6 Integration: Why?

IPv4 address space depletion

- IANA pool already depleted (Feb. 2011)
- Projection on RIR pool depletion: 2011-2012 (per RIR basis)
- LIRs' pools will be depleted later...
- New companies will not be able to get IPv4 address space
- Existing companies will not be able to extend theirs

Complexity increasing in the IPv4 world (networks & services):

- Lack/absence of routable IPv4 addresses
- NAT violates the "end-to-end" principle, multiple-level NATs coming (NAT444)!
- Even private space (RFC 1918) is not enough for some networks (example: Comcast would need 100 M+ @ to address their subscribers' set-top boxes)
- NAT Traversal development cost is getting unbearable

Why Integration?

- IPv4 and IPv6 are incompatible
  - Different packet format
  - Prefixes are different
- No backward compatibility, but management is very similar.
- IETF planned to deploy IPv6 then make IPv4 disappeared
  - but Metcalf's law was on IPv4 side.
  - Content on IPv4, so few actors moved.
  - Not a complete chain so access is difficult.
- Some Integration mechanisms are dangerous

Easy integration ? Not completely true

- OSes have integrated IPv6
  - Window 7, iOS, Linux,. . .
- Some applications are compatible with IPv6
  - see http://en.wikipedia.org/wiki/Comparison_of_IPv6_application_support
- Routers have integrated IPv6
  - Cisco, Juniper, ALU,. . .
- but the chain is not complete, so IPv6 is not fully available
- An address is not only used to forward packet
  - Allocation procedures
  - Management (size is different)
  - . . .
- IPv6 is new. Test products before production!

Integration

6 generic scenarios

Communications Model

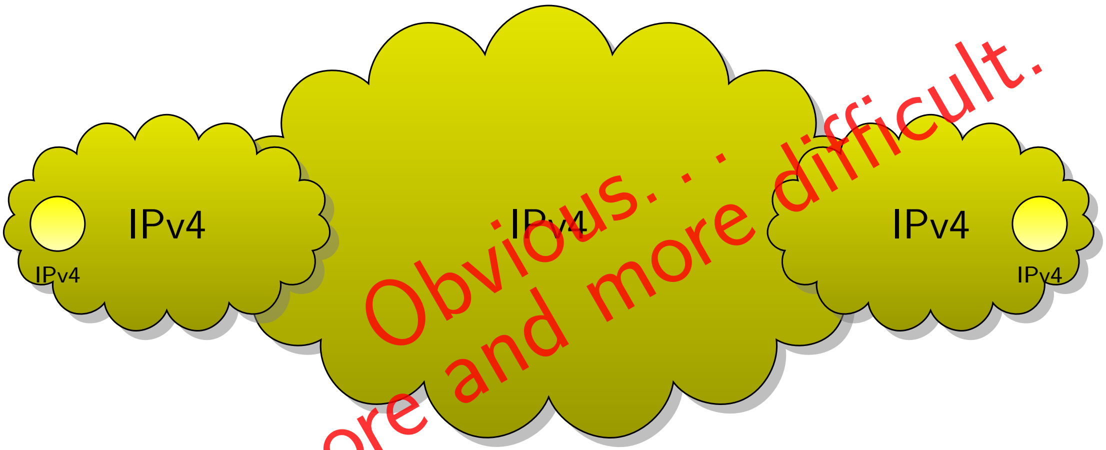
Who initiates communication towards whom (6 possibilities)?

1. An IPv4 system connects to an IPv4 system through an IPv4 network
2. An IPv6 system connects to an IPv6 system through an IPv6 network
3. An IPv4 system connects to an IPv4 system through an IPv6 network
4. An IPv6 system connects to an IPv6 system through an IPv4 network
5. An IPv4 system connects to an IPv6 system
6. An IPv6 system connects to an IPv4 system

Complexity

- 1) & 2) : Quite obvious
- 3) & 4) : Less easy but no real problem
- 5) & 6) : Quite complex. There is no global solution today (different partial solutions following different approaches)

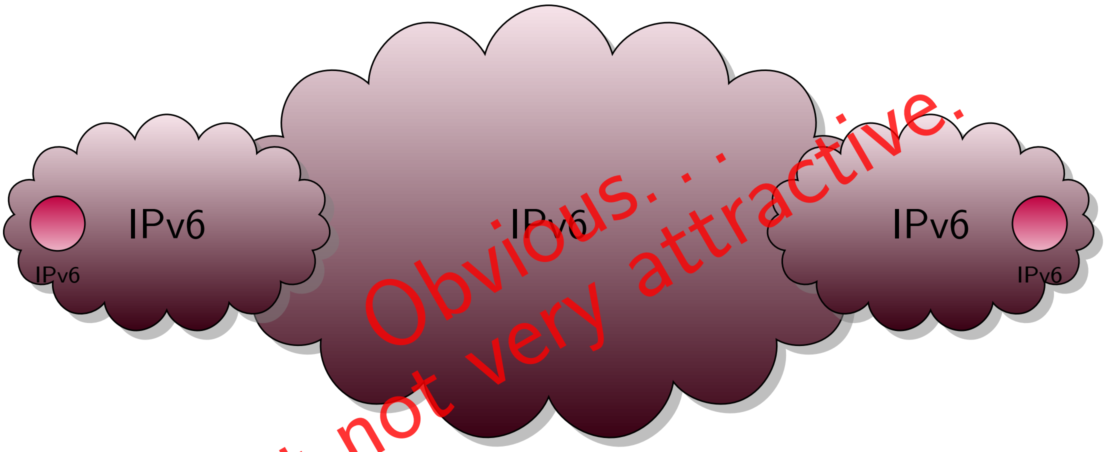
An IPv4 system connects to an IPv4 system through an IPv4 network



Obvious. . .

But more and more difficult.

An IPv6 system connects to an IPv6 system through an IPv6 network

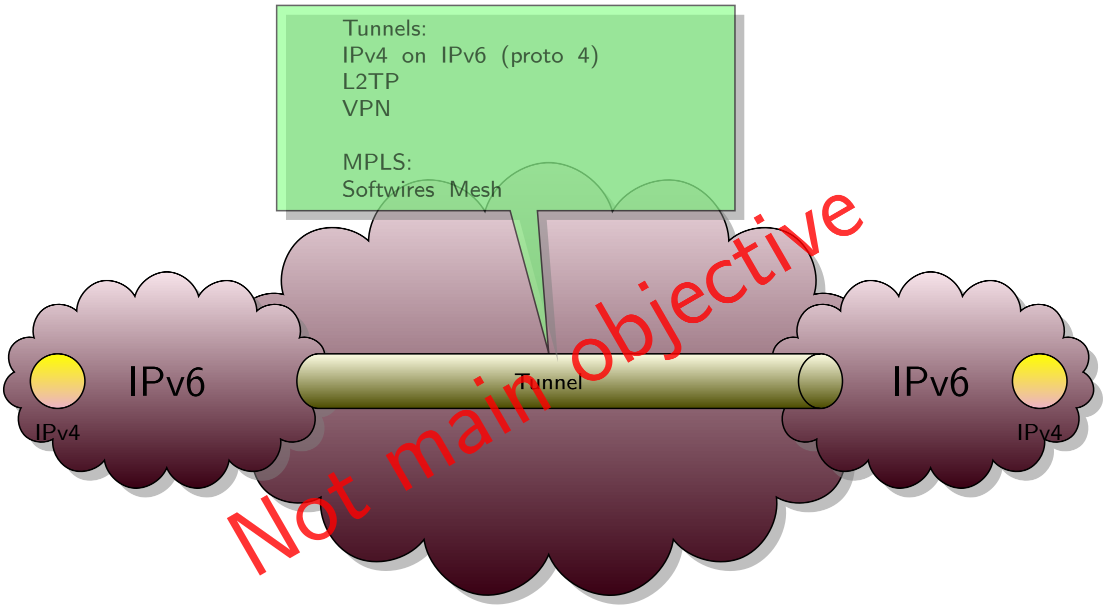


Obvious. . .

But not very attractive.

An IPv4 system connects to an IPv4 system through an IPv6 network

Integration
Why IPv6 Integration ?
6 generic scenarios
Tools overview
Scenarios
Backbone operator
Internet Access Provider
3G/LTE
Enterprise
Home network and SOHO
Programming IPv6 Applications



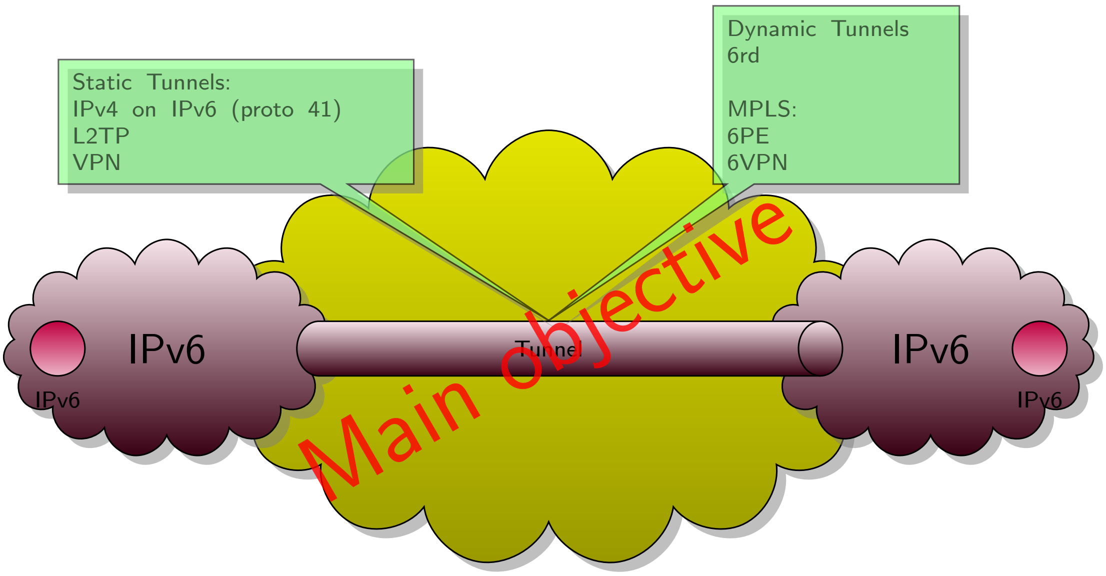
Tunnels:
IPv4 on IPv6 (proto 4)
L2TP
VPN

MPLS:
Softwires Mesh

Not main objective

An IPv6 system connects to an IPv6 system through an IPv4 network

Integration
Why IPv6 Integration ?
6 generic scenarios
Tools overview
Scenarios
Backbone operator
Internet Access Provider
3G/LTE
Enterprise
Home network and SOHO
Programming IPv6 Applications



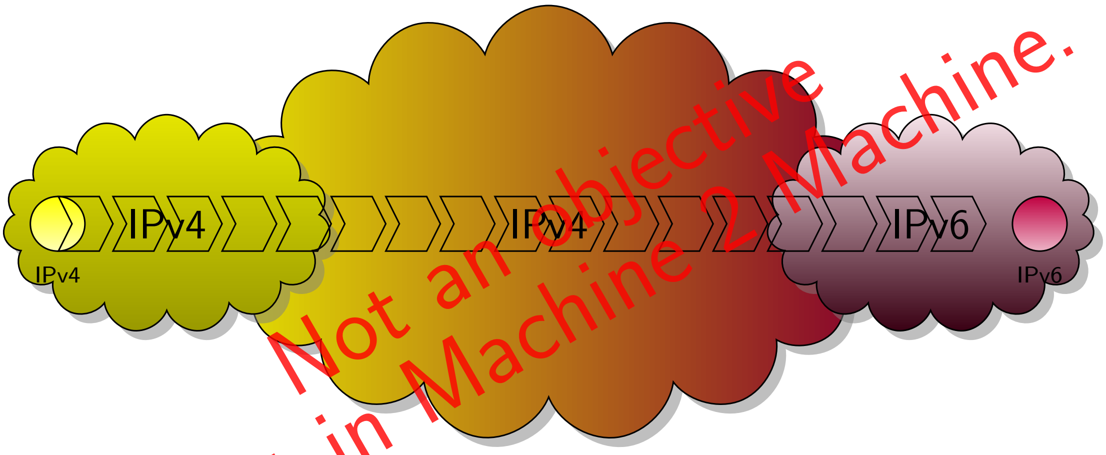
Static Tunnels:
IPv4 on IPv6 (proto 41)
L2TP
VPN

Dynamic Tunnels
6rd

MPLS:
6PE
6VPN

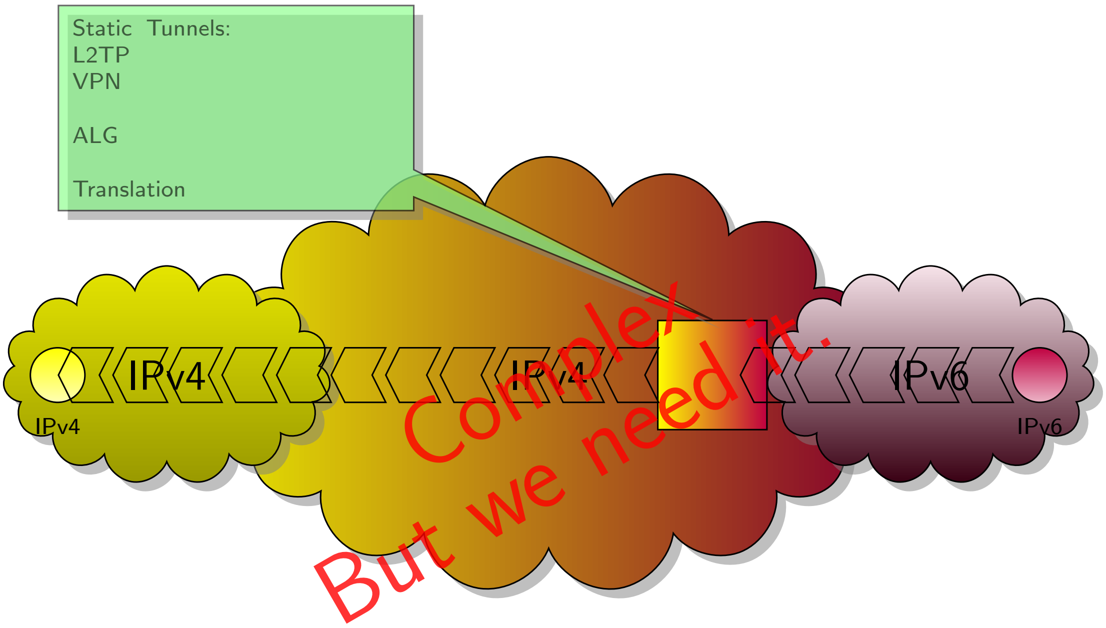
Main objective

An IPv4 system connects to an IPv6 system



Not an objective
Except in Machine 2 Machine.

An IPv6 system connects to an IPv4 system



Static Tunnels:
L2TP
VPN

ALG

Translation

Complex
But we need it.

Integration

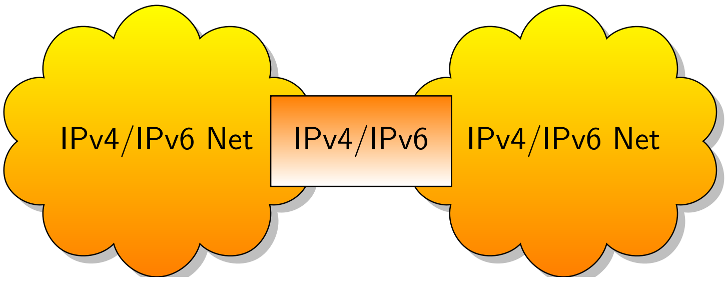
Tools overview

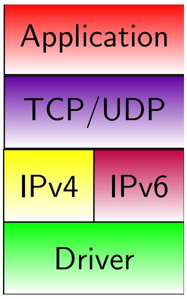
Rough Classification of Transition/Integration Mechanisms

- v6-v6 or v4-v4 Communication
  - Dual-Stack: v4 and v6 are fully available end-to-end
- Tunneling
  - v4 communication through a v6 network or vice versa
  - **automatic** vs **configured** (manual) tunnels
- v4-v6 co-existence/cross-communication
  - Translation
    - Header / protocol / port (v6→v4 and v4→v6)
    - Stateless *vs* Stateful
  - Relays / Application Level Gateways (ALG)

Dual-Stack Approach (RFC 4213)

- IPv4 and IPv6 running on the same box
- Especially useful for "Legacy" (existing) networks
  - *V6-fied* (legacy) IPv4 servers can provide the same service over IPv6 transport for new IPv6-only clients (web, mail, ftp, ssh. . .)
  - *V6-fied* (legacy) IPv4 clients can query new IPv6-only servers



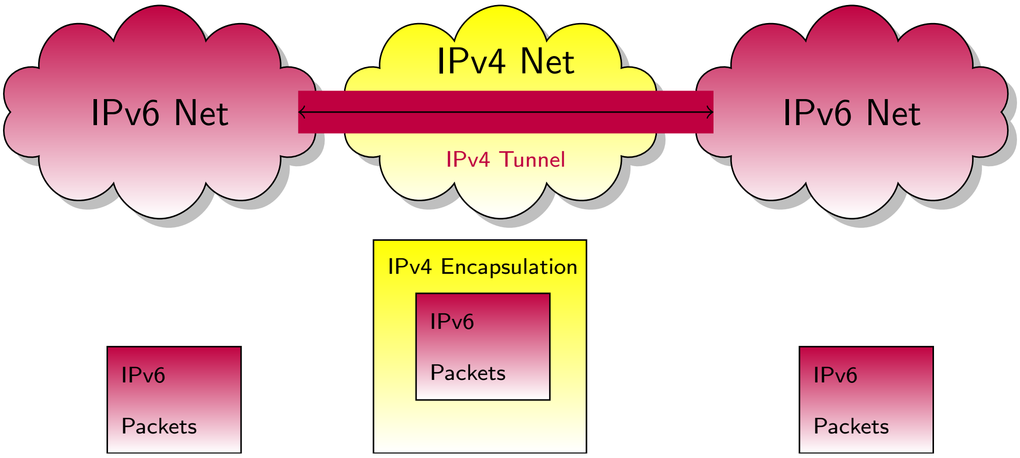


- But. . .
  - At least one IPv4 address is required for every node
  - ⇒ Alone, this approach does not fix the issue of IPv4 space exhaustion!
  - ⇒ Need to manage both protocols

Generic Approach for "Tunneling"

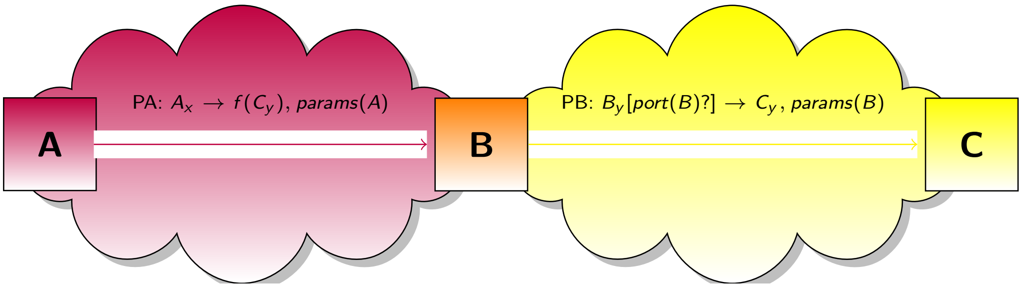
2 types of tunnels:

- Automatic Tunnels
  - Examples : 6to4, Teredo, ISATAP, 6PE/MPLS. . .
- Configured Tunnels
  - Manual, "Tunnel Broker"
- IP on IP cannot be NATed



Generic Approach for "Translation"

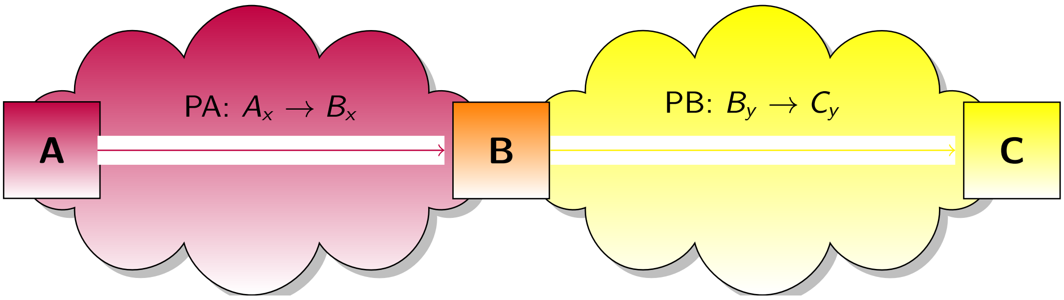
Integration
Why IPv6 Integration ?
6 generic scenarios
Tools overview
Scenarios
Backbone operator
Internet Access Provider
3G/LTE
Enterprise
Home network and SOHO
Programming IPv6 Applications


A
PA: $A_x \to f(C_y), params(A)$
B
PB: $B_y[port(B)?] \to C_y, params(B)$
C

- $(x, y) \in \{(6, 4), (4, 6)\}$
- A is IPv_x-only, C is IPv_y-only
- A sends a packet PA to C
  - Source address: A_x
  - Destination address: $C_x = f(C_y)$ (an IPv_x mapped to C_y)
- Packet PA is intercepted by B, the translation box supporting both IPv_x and IPv_y
- Packet PA is translated into packet PB, later sent to C
  - Source address: B_y from the "shared pool", potentially with a new port(B)
  - Destination address: C_y

Generic Approach for ALGs ("proxy")

Integration
Why IPv6 Integration ?
6 generic scenarios
Tools overview
Scenarios
Backbone operator
Internet Access Provider
3G/LTE
Enterprise
Home network and SOHO
Programming IPv6 Applications



- $(x, y) \in \{(6, 4), (4, 6)\}$
- A is an IPv_x-only client; C is IPv_y-only server
- A sends to B a packet PA containing a request targeting C
  - Source address: A_x
  - Destination address: B_x
- B is a proxy supporting both IPv_x and IPv_y
- B sends to C a **new packet** PB, *proxying* A?s request
  - Source address: B_y
  - Destination address: C_y
- Examples: proxy web/ftp/DNS/mail...

Integration

Scenarios

Where to act, what to do exactly?

- For ISPs/Operators
  - Backbone routers, Border routers (peering, transit)
    - Performances, Management
  - Access equipment (wired or wireless)
    - Prefix Allocation
- For users (individuals, enterprise, campus. . .):
  - LAN (routers if any)
  - Firewalls
  - Connectivity (CPE, PE)
  - Getting through their v4 ISP or bypassing it
- For everybody:
  - OS (local and distant)
  - Network applications or applications invoking the network even transiently

IPv6 is not mandatory everywhere to start Integration

Integration

Backbone operator

Backbone operators

- Forward IPv6 as fast as IPv4
- Some old routers forward IPv6 in the supervision card
  - bad performances
- Tunnel is not a good solution
  - bad performances due to encapsulation
- MPLS is your friend.
  - L2VPN
  - 6PE
  - 6VPN
- Few have the opposite problem:
  - How to carry IPv4 traffic on an IPv6 backbone
  - Softwires mesh

Integration

Internet Access Provider

ISP

- Performances in forwarding (not so strict)
  - may use tunnels
- Allocate IPv6 prefixes
  - Lawfull IP address identification.
- May suffer from IPv4 shortage
- Different strategies exist

Define an addressing plan (Renater case study)

↓RIPE-NCC

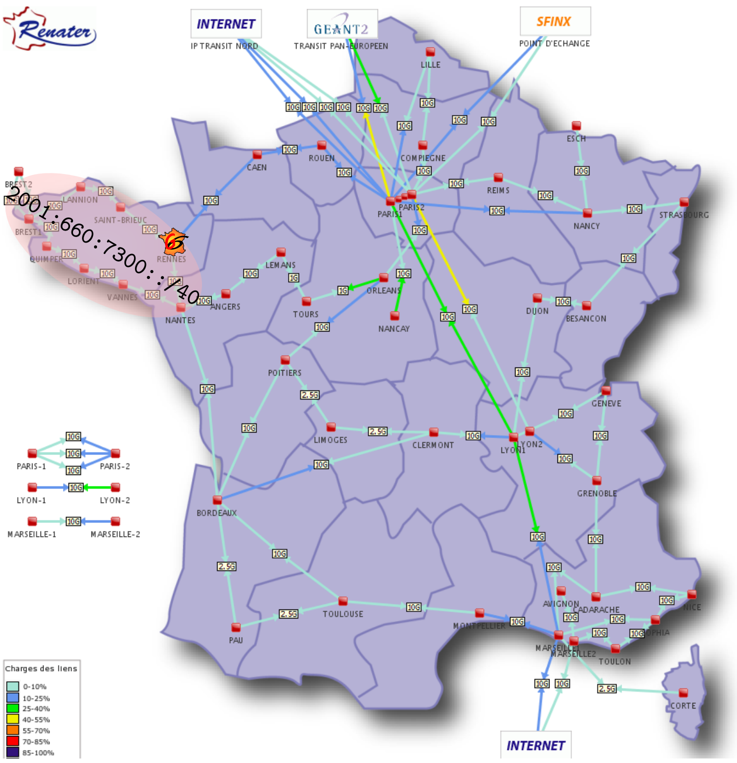
2001:660::/32

↓POP

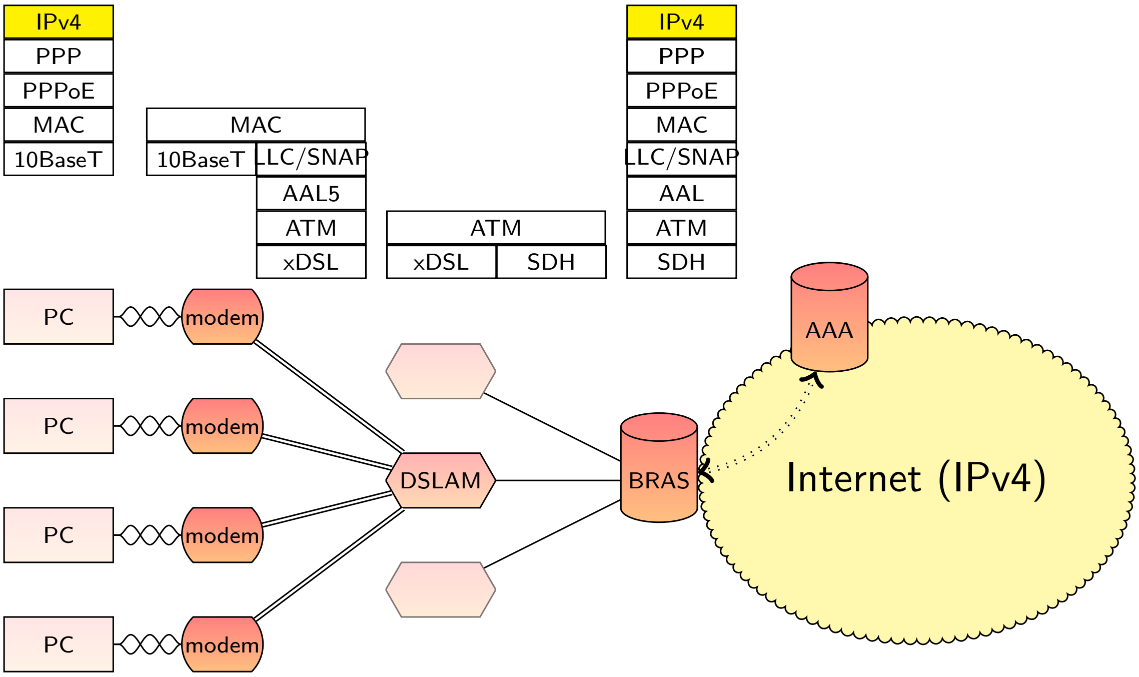
2001:660:7300::/40

↓Site

2001:660:7301::/48

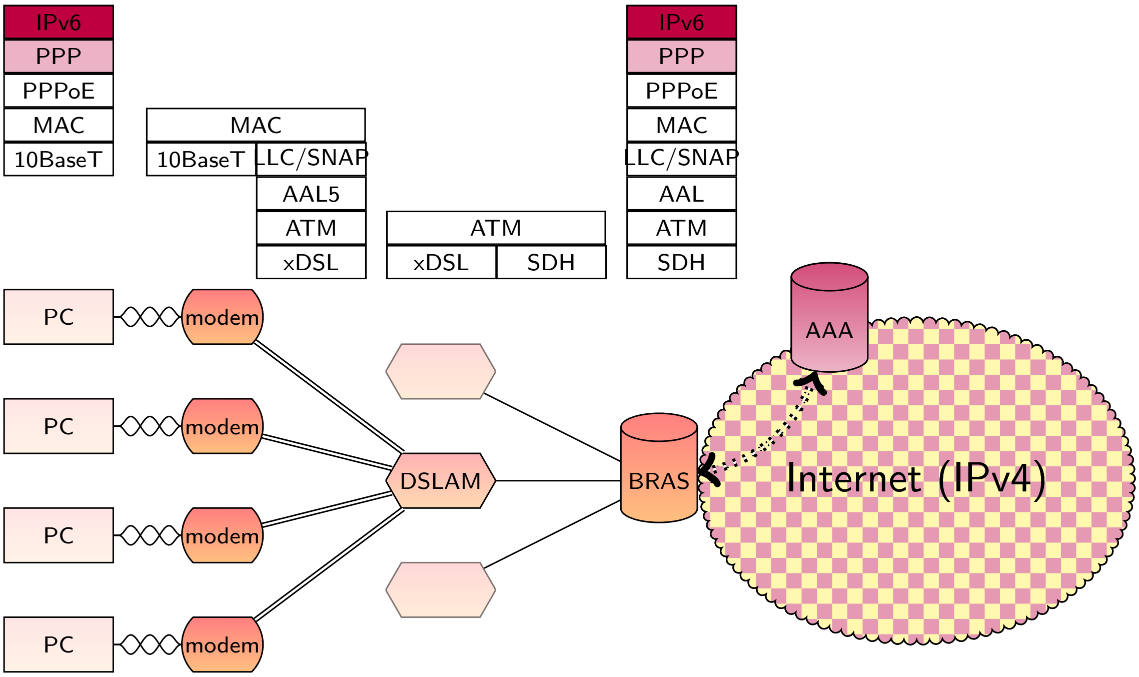


ADSL Architecture



ADSL Architecture

Integration
Why IPv6 Integration ?
6 generic scenarios
Tools overview
Scenarios
Backbone operator
Internet Access Provider
3G/LTE
Enterprise
Home network and SOHO
Programming IPv6 Applications



IPv6
PPP
PPPoE
MAC
10BaseT

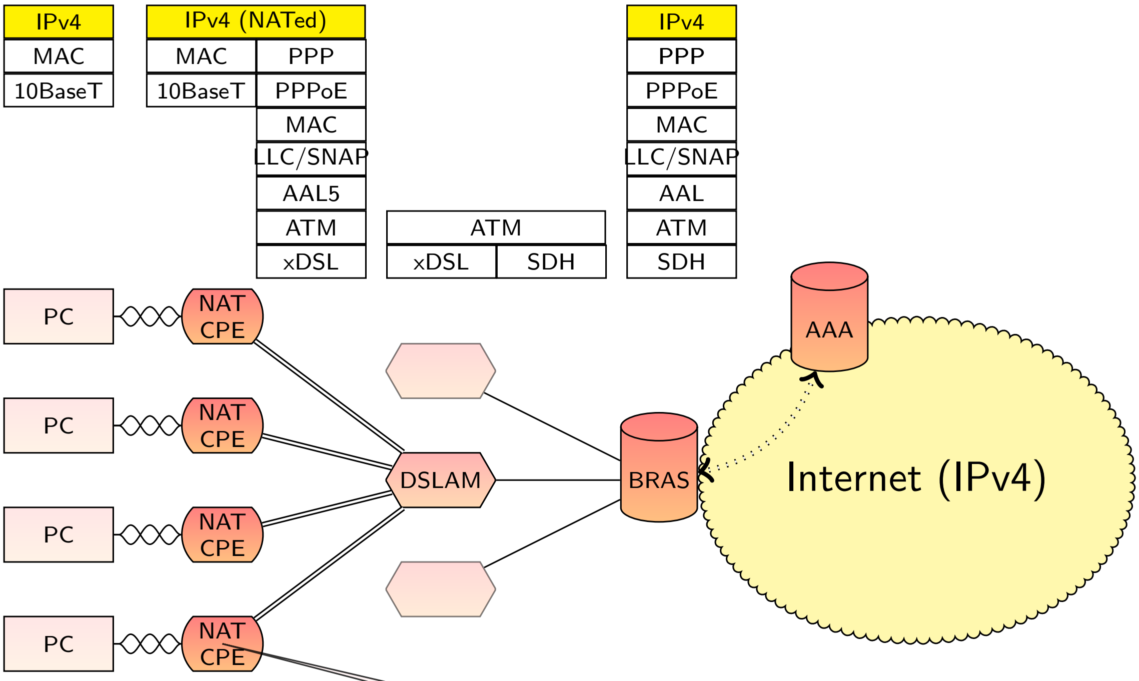
MAC
10BaseT | LLC/SNAP
AAL5
ATM
xDSL

ATM
xDSL | SDH

IPv6
PPP
PPPoE
MAC
LLC/SNAP
AAL
ATM
SDH

ADSL Architecture (Box or CPE)

Integration
Why IPv6 Integration ?
6 generic scenarios
Tools overview
Scenarios
Backbone operator
Internet Access Provider
3G/LTE
Enterprise
Home network and SOHO
Programming IPv6 Applications



IPv4
MAC
10BaseT

IPv4 (NATed)
MAC | PPP
10BaseT | PPPoE
MAC
LLC/SNAP
AAL5
ATM
xDSL

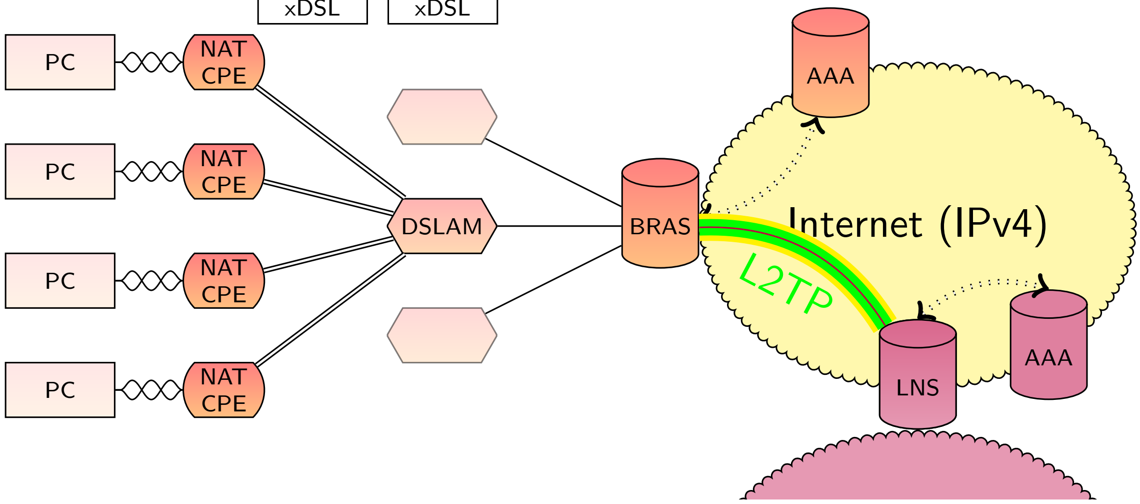
ATM
xDSL | SDH

IPv4
PPP
PPPoE
MAC
LLC/SNAP
AAL
ATM
SDH

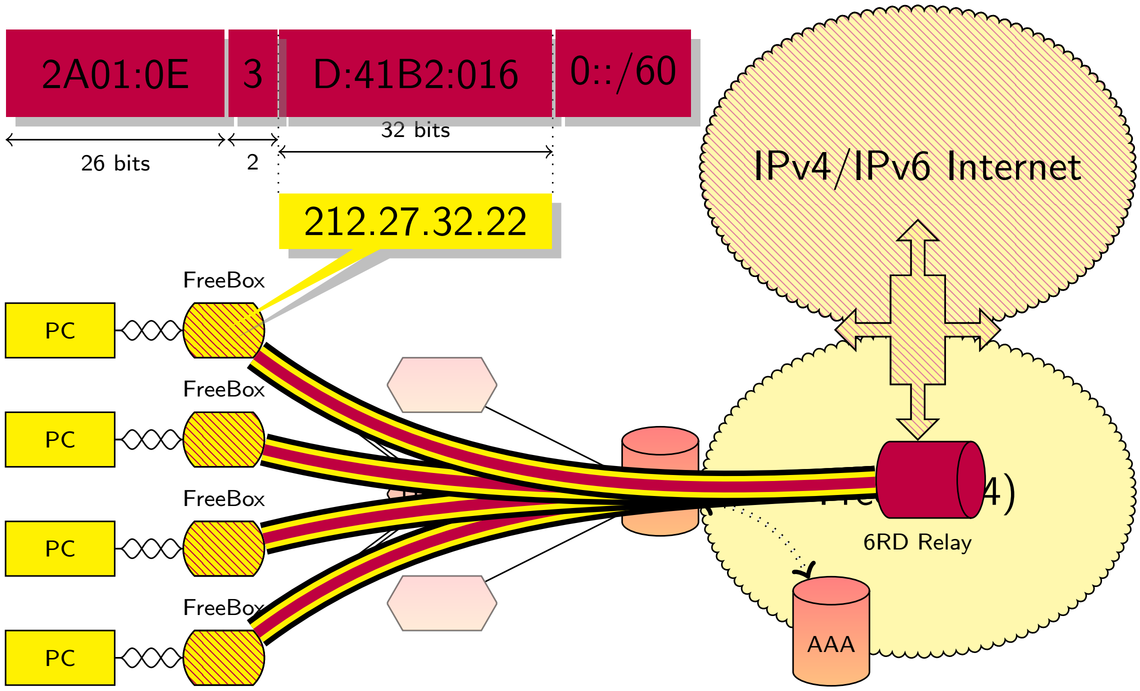
Must be changed or upgraded

ADSL Architecture (3rd Generation DSLAM)

IPv4	IPv4 (NATed)		IPv4		IPv4
MAC	MAC	PPP	PPPE	PPP	PPP
10BaseT	10BaseT	PPPoE	PPPoE	SDH	SDH
		MAC	MAC		
		LLC/SNAP	LLC/SNAP		
		AAL5	AAL5		
		ATM	ATM		
		xDSL	xDSL		



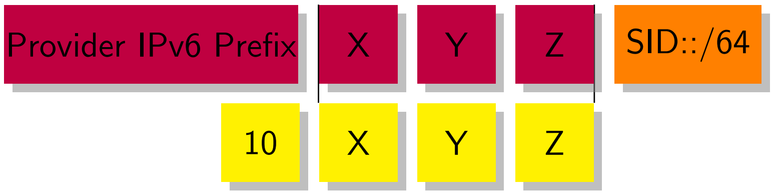
Free - 6rd (RFC 5969)



6rd

Integration
Why IPv6 Integration ?
6 generic scenarios
Tools overview
Scenarios
Backbone operator
Internet Access Provider
3G/LTE
Enterprise
Home network and SOHO
Programming IPv6 Applications

- Core network or DSLAM are not changed:
  - only some 6RD relays and CPE modification.
- IPv6 prefixes are stable if IPv4 addresses are stable
- No need to manage/log IPv6 prefixes since IPv4 prefix is embedded
- 6RD relay is not used for internal traffic
- Deployed in Free Network in 2007 in 5 weeks.
- DHCPv4 option to setup 6RD relays (6RD Relays, and prefix lengths)
- Can work with IPv4 private addresses.

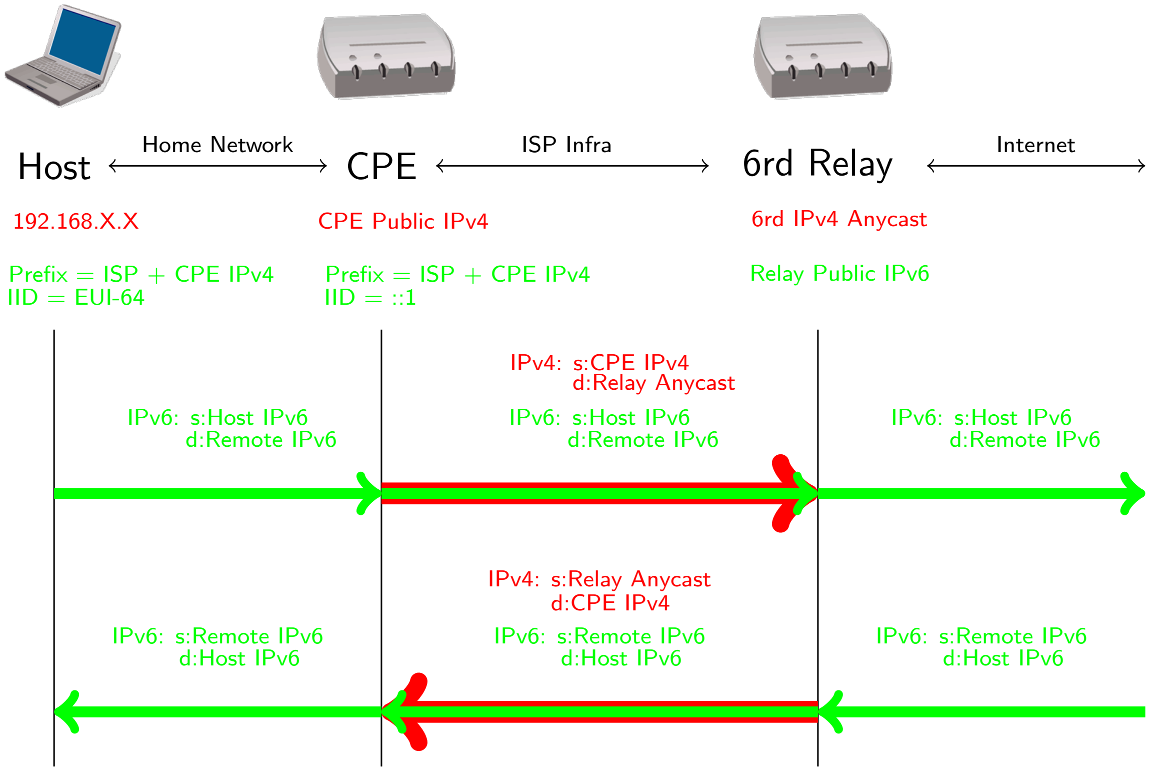


Provider IPv6 Prefix | X | Y | Z | SID::/64

10 | X | Y | Z

6rd: Mechanism

Integration
Why IPv6 Integration ?
6 generic scenarios
Tools overview
Scenarios
Backbone operator
Internet Access Provider
3G/LTE
Enterprise
Home network and SOHO
Programming IPv6 Applications



Host ← Home Network → CPE ← ISP Infra → 6rd Relay ← Internet →

Host: 192.168.X.X
Prefix = ISP + CPE IPv4
IID = EUI-64

CPE: CPE Public IPv4
Prefix = ISP + CPE IPv4
IID = ::1

6rd Relay: 6rd IPv4 Anycast
Relay Public IPv6

IPv6: s:Host IPv6 d:Remote IPv6

IPv4: s:CPE IPv4 d:Relay Anycast
IPv6: s:Host IPv6 d:Remote IPv6

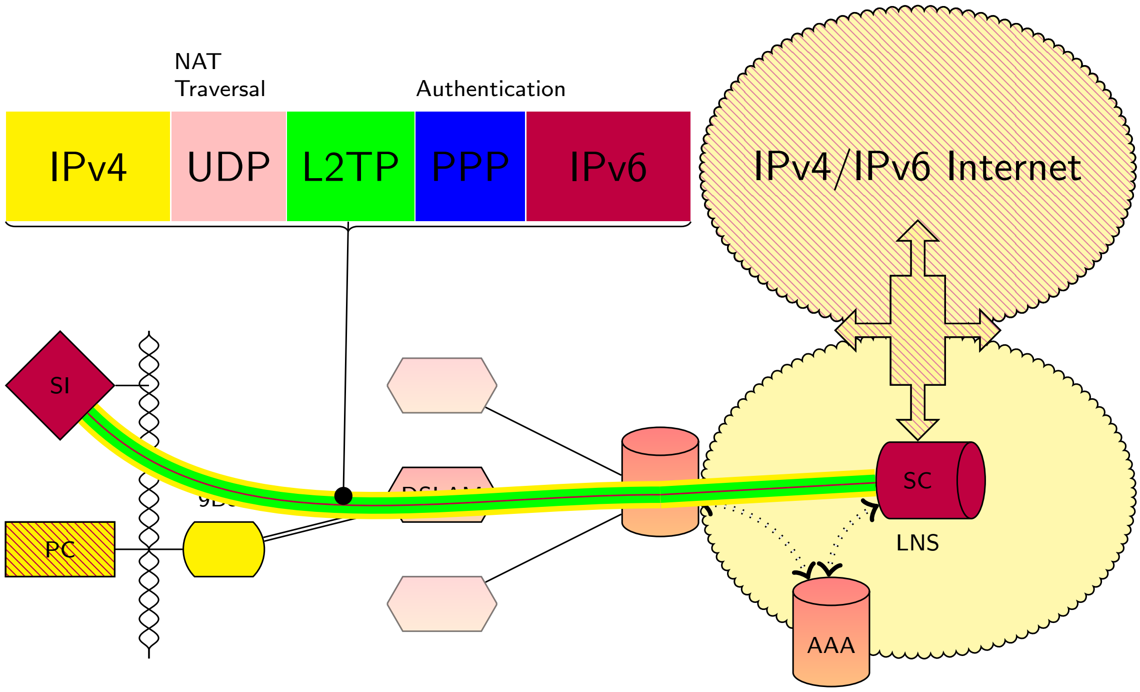
IPv6: s:Host IPv6 d:Remote IPv6

IPv6: s:Remote IPv6 d:Host IPv6

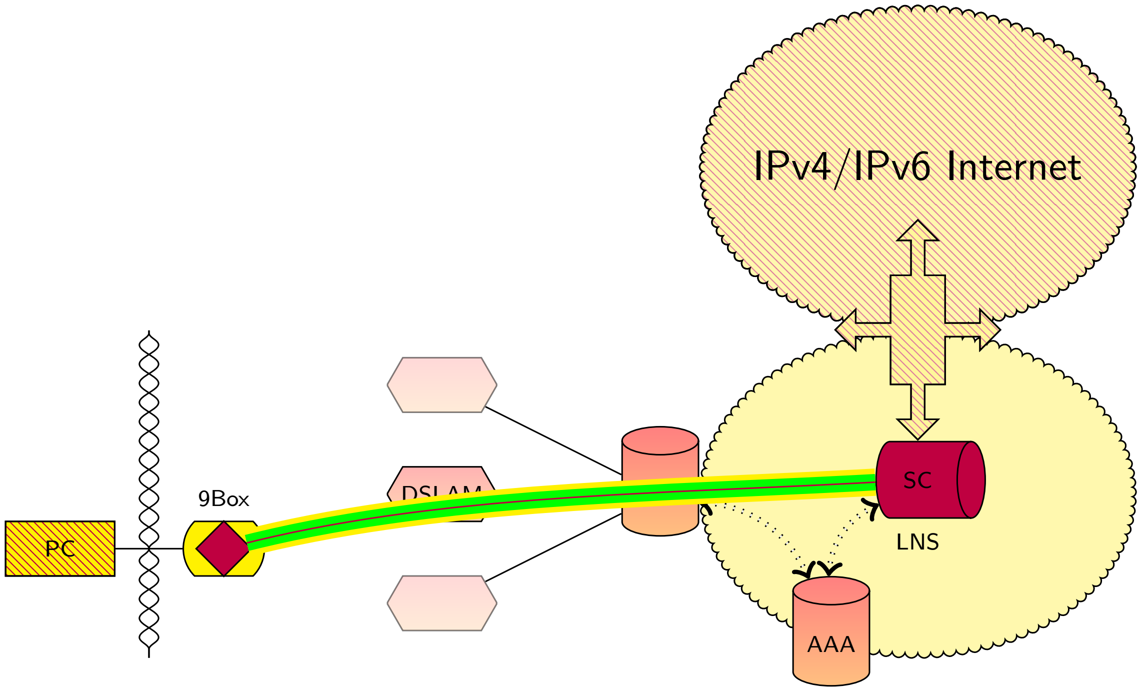
IPv4: s:Relay Anycast d:CPE IPv4
IPv6: s:Remote IPv6 d:Host IPv6

IPv6: s:Remote IPv6 d:Host IPv6

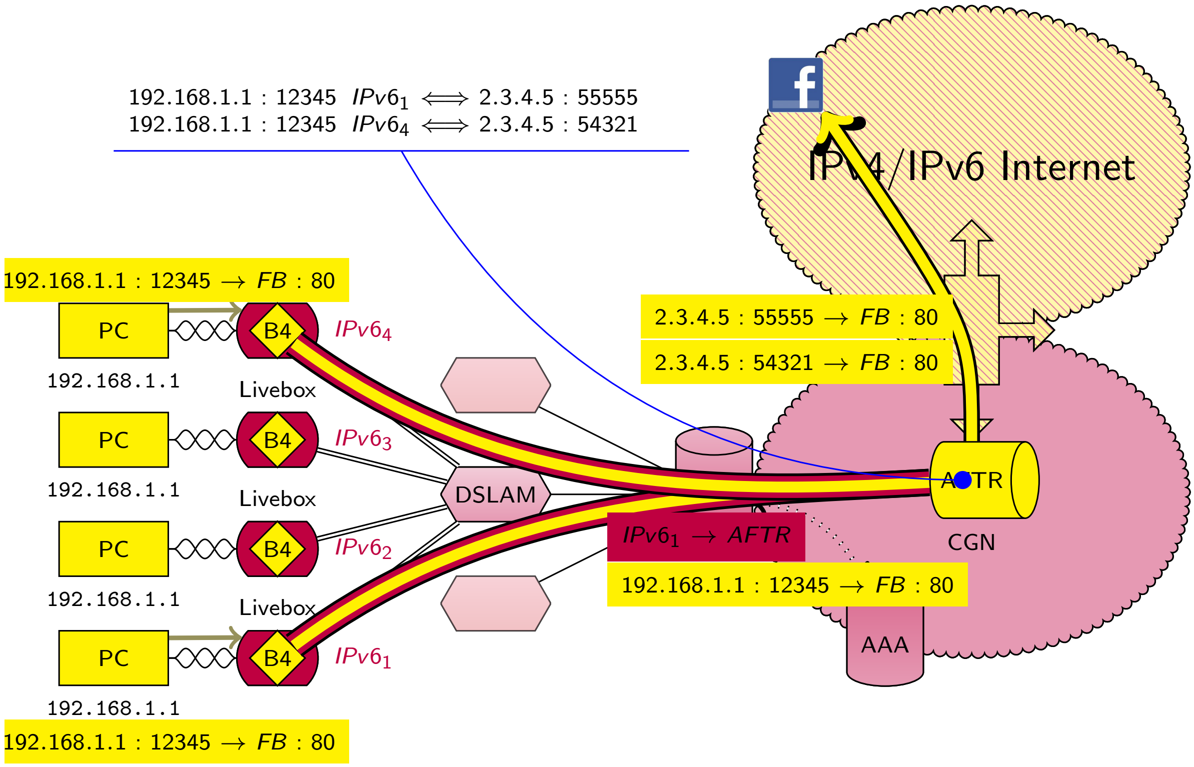
SFR: Softwires: H&S Architecture RFC 5571



SFR: Softwires: H&S Architecture RFC 5571



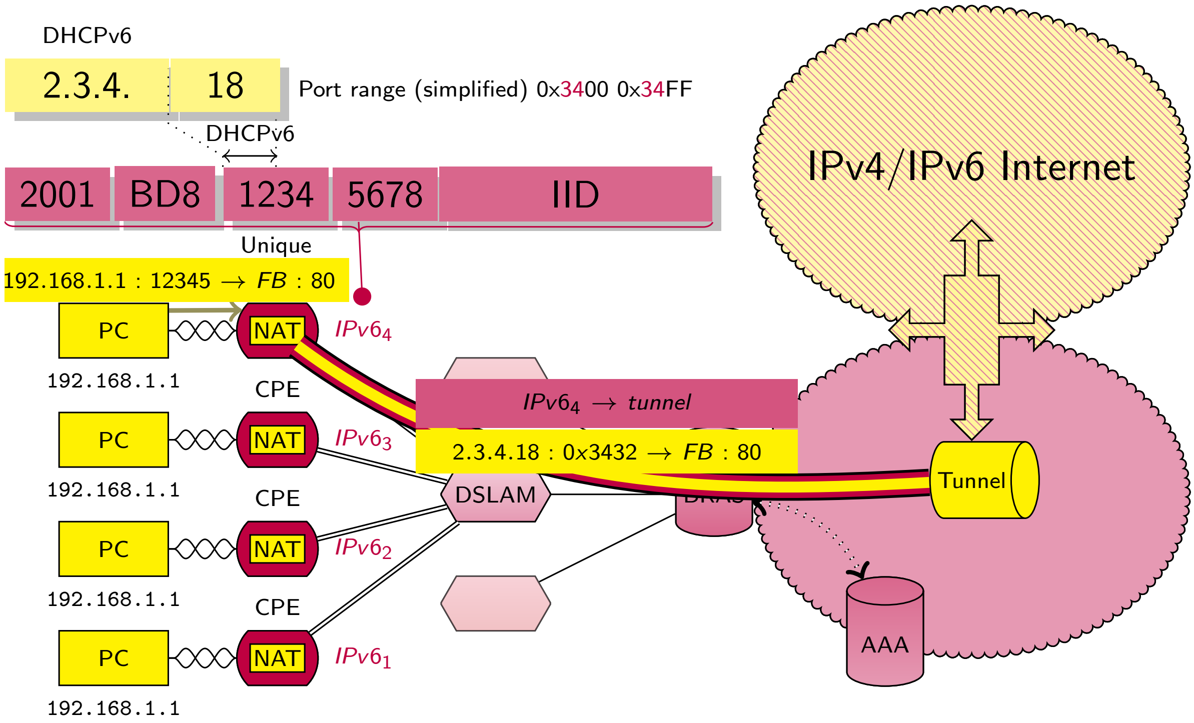
France Telecom/Orange: Native + CGN


192.168.1.1 : 12345 $IPv6_1$ ⟺ 2.3.4.5 : 55555
192.168.1.1 : 12345 $IPv6_4$ ⟺ 2.3.4.5 : 54321
IPv4/IPv6 Internet
192.168.1.1 : 12345 → FB : 80
PC
B4
$IPv6_4$
192.168.1.1
Livebox
PC
B4
$IPv6_3$
192.168.1.1
Livebox
DSLAM
PC
B4
$IPv6_2$
192.168.1.1
Livebox
PC
B4
$IPv6_1$
192.168.1.1
192.168.1.1 : 12345 → FB : 80
2.3.4.5 : 55555 → FB : 80
2.3.4.5 : 54321 → FB : 80
AFTR
CGN
$IPv6_1$ → $AFTR$
192.168.1.1 : 12345 → FB : 80
AAA

France Telecom/Orange: Native + CGN

- Carrier Grade NAT deals with IPv4 address exhaustion:
  - No IPv4 address for the infrastructure
  - An IPv4 address is shared among several users
    - A user consumes about 300 port numbers
    - Less is needed (2 or 3 users per address)
- Less scalable than user NAT
  - More traffic from different users
  - for incoming traffic must map a port number to an IPv6 address
- Must take into account:
  - UPnP: Send UPnP traffic to CGN (see Port Control Protocol)
  - Static Mapping: Web page on AFTER
- Legal identification is complex:
  - Log per flow
  - Need IPv4 address, port number and time.

4rd (main idea)



DHCPv6

2.3.4. | 18

Port range (simplified) 0x3400 0x34FF

DHCPv6

2001 | BD8 | 1234 | 5678 | IID

Unique

192.168.1.1 : 12345 → *FB* : 80

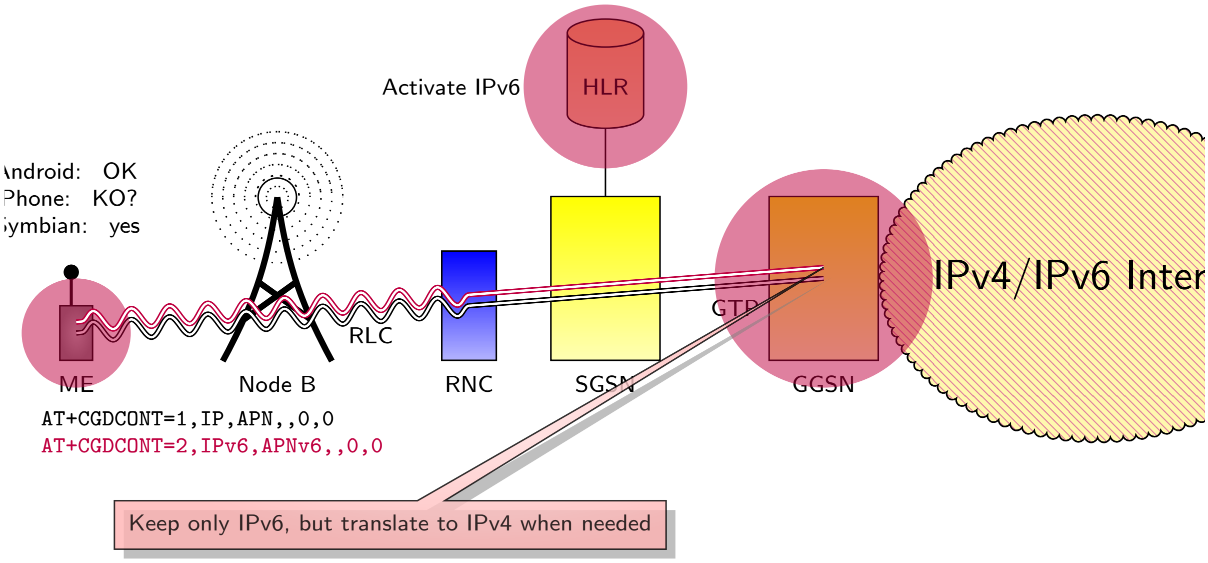
IPv6$_4$ → *tunnel*

2.3.4.18 : 0x3432 → *FB* : 80

Integration

3G/LTE

3G data

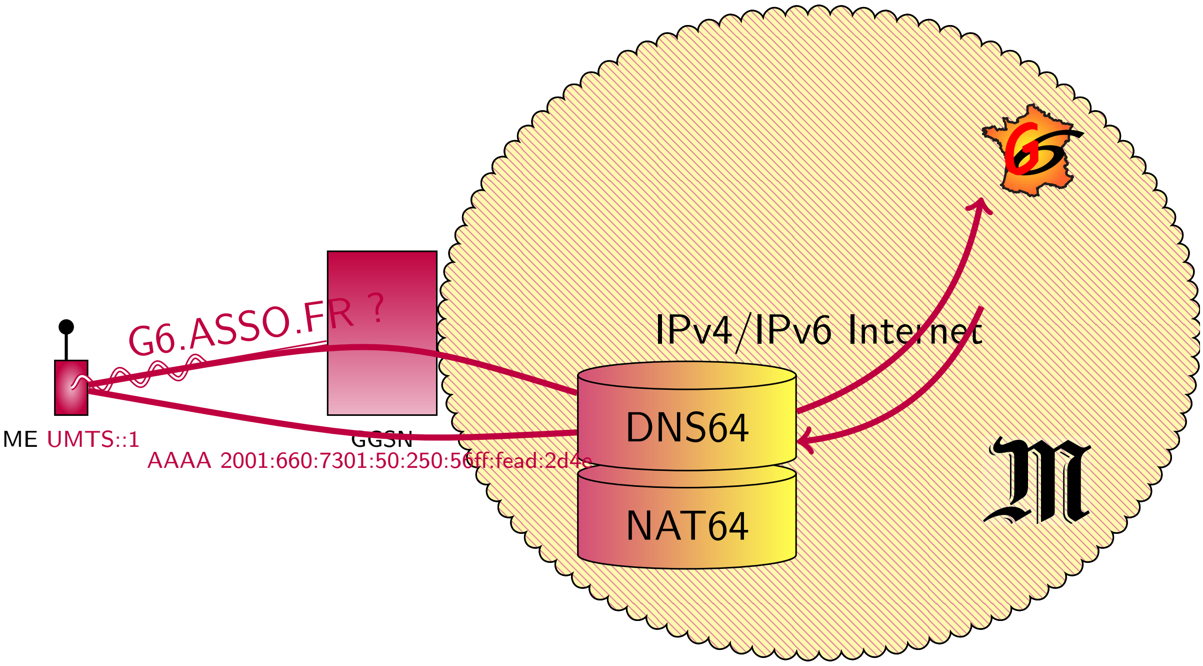


```
AT+CGDCONT=1,IP,APN,,0,0
AT+CGDCONT=2,IPv6,APNv6,,0,0
```

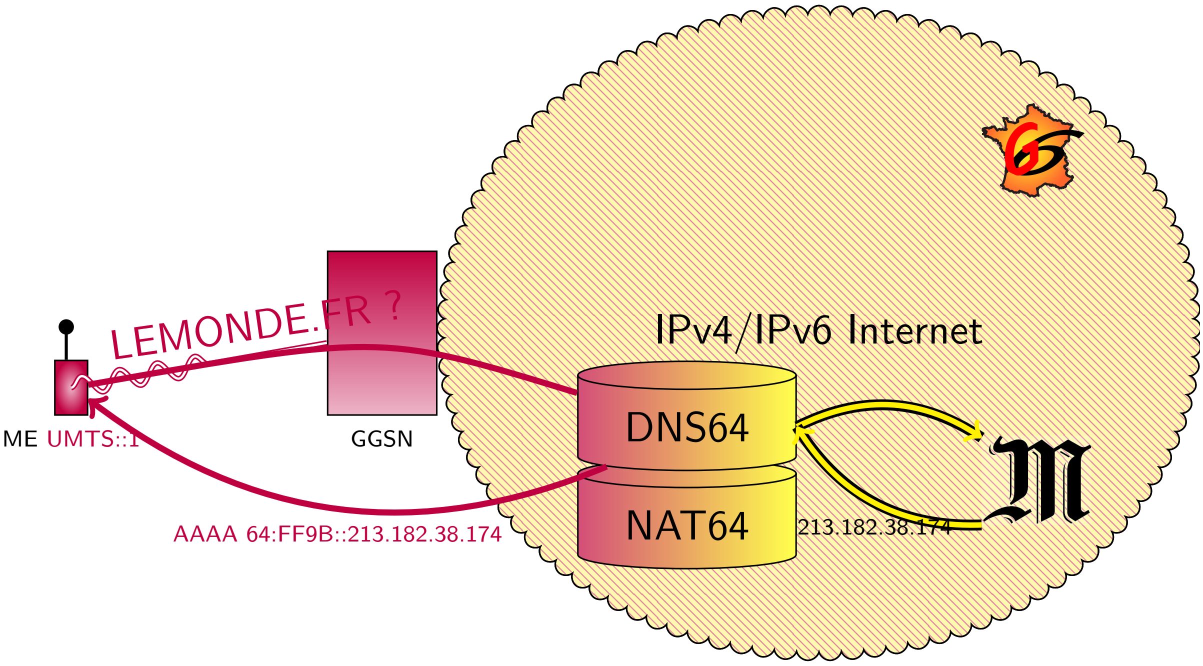
Keep only IPv6, but translate to IPv4 when needed

ME: Mobile Equipment, **RNC**: Radio Network Controller, **SGSN**: Serving GPRS Support Node, **GGSN**: Gateway GPRS Support Node, **HLR**: Home Location Register, **GTP**: GPRS Tunnelling Protocol **RLC**: Radio Link Control

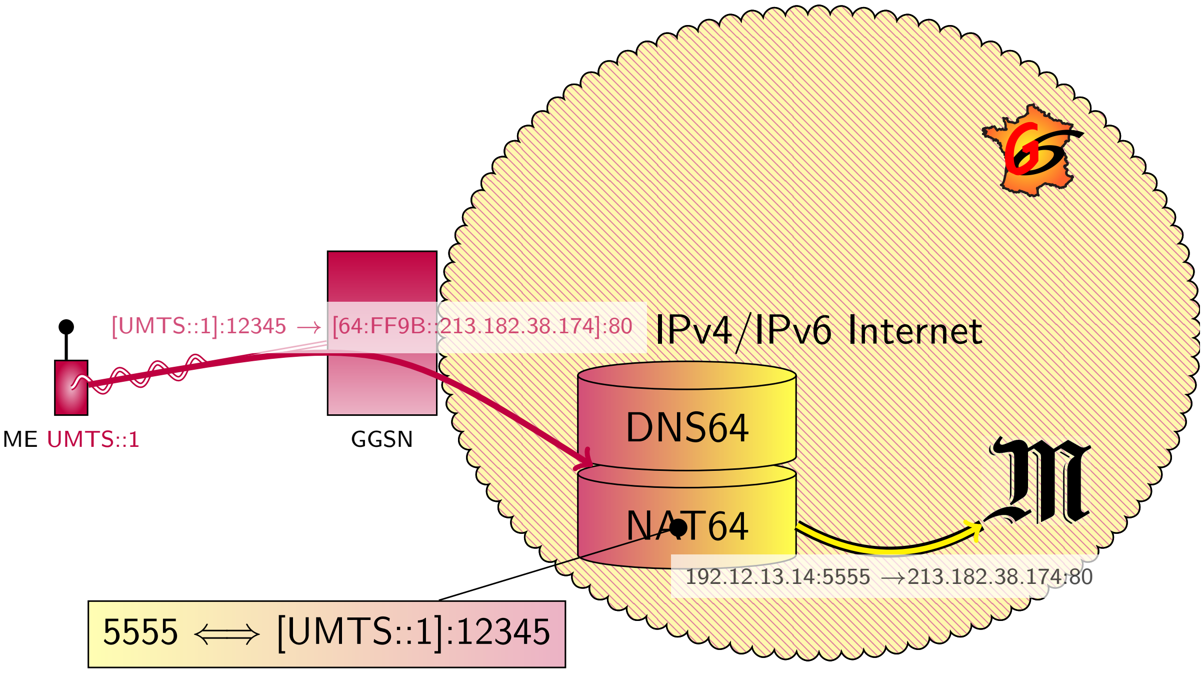
3G data + NAT64/DNS64



3G data + NAT64/DNS64



3G data + NAT64/DNS64



Integration

Enterprise

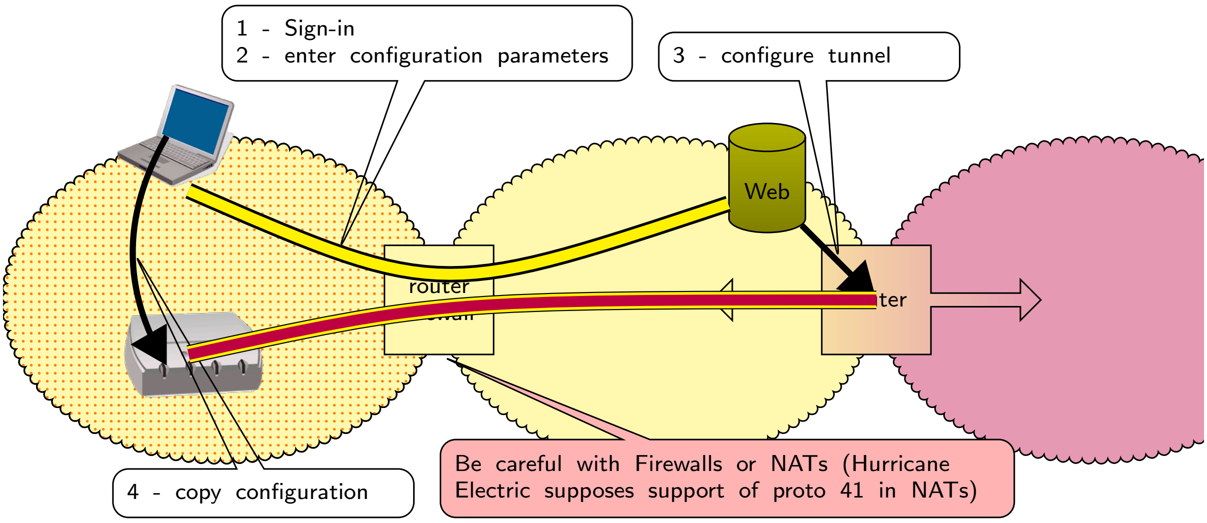
Entreprise Network

- Anticipate: include IPv6 in calls for tenders.
  - RIPE 501 is your friend (W http://www.ripe.net/ripe/docs/ripe-501)
- Define your goal:
  - Test: learn about IPv6 or develop products
    - Get temporary connectivity (Tunnel Brokers)
  - V6fy Extranet or/and Intranet
    - Get permanent connectivity and prefix
    - Define addressing plan
    - Define security rules

Tunnel Broker (RFC 3053)

- Hurricane Electric (tunnelbroker.com)
  - Standard and BGP tunnels
  - Point of Presence in Asia, North America and Europe
- sixxs (http://www.sixxs.net/main/)
  - Worldwide
- gogo6 (http://gogonet.gogo6.com/page/freenet6-tunnelbroker)
  - Few Point of Presence
  - in Canada
  - NAT Traversal

Tunnel Brokers



1 - Sign-in
2 - enter configuration parameters

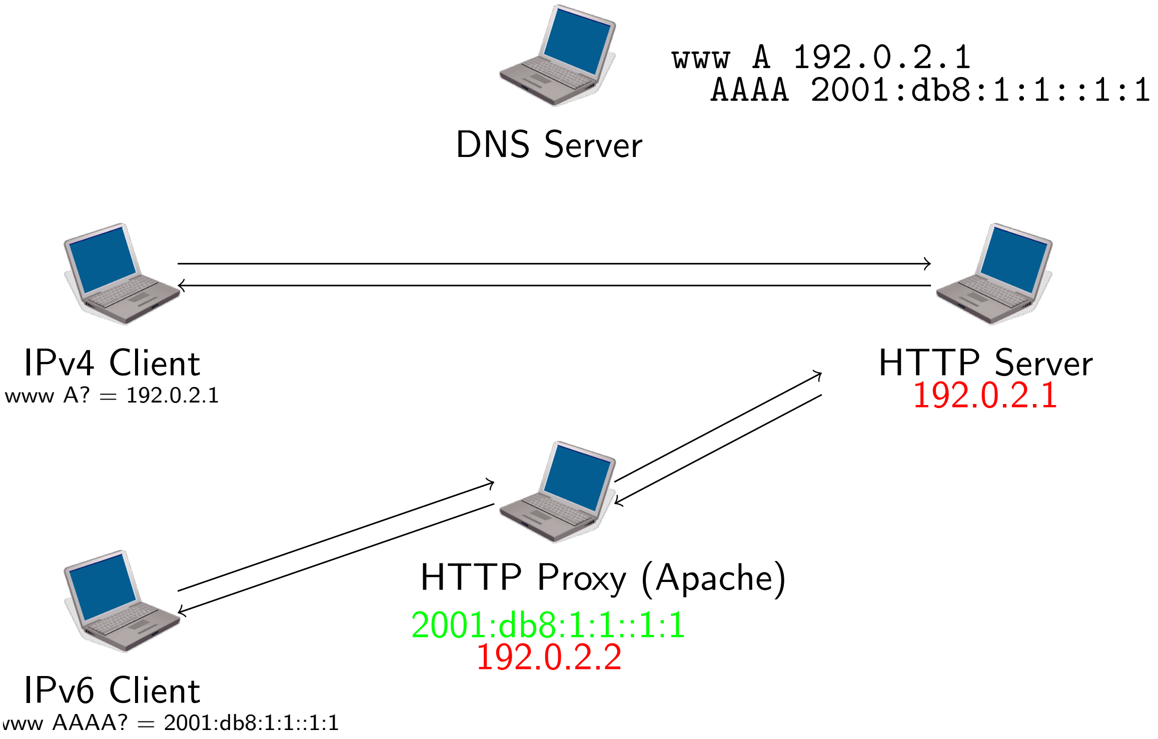
3 - configure tunnel

4 - copy configuration

Be careful with Firewalls or NATs (Hurricane Electric supposes support of proto 41 in NATs)

Application Level Gateway

How to enable IPv6 access to a production Web site



```
www A 192.0.2.1
  AAAA 2001:db8:1:1::1:1
```

DNS Server

IPv4 Client
www A? = 192.0.2.1

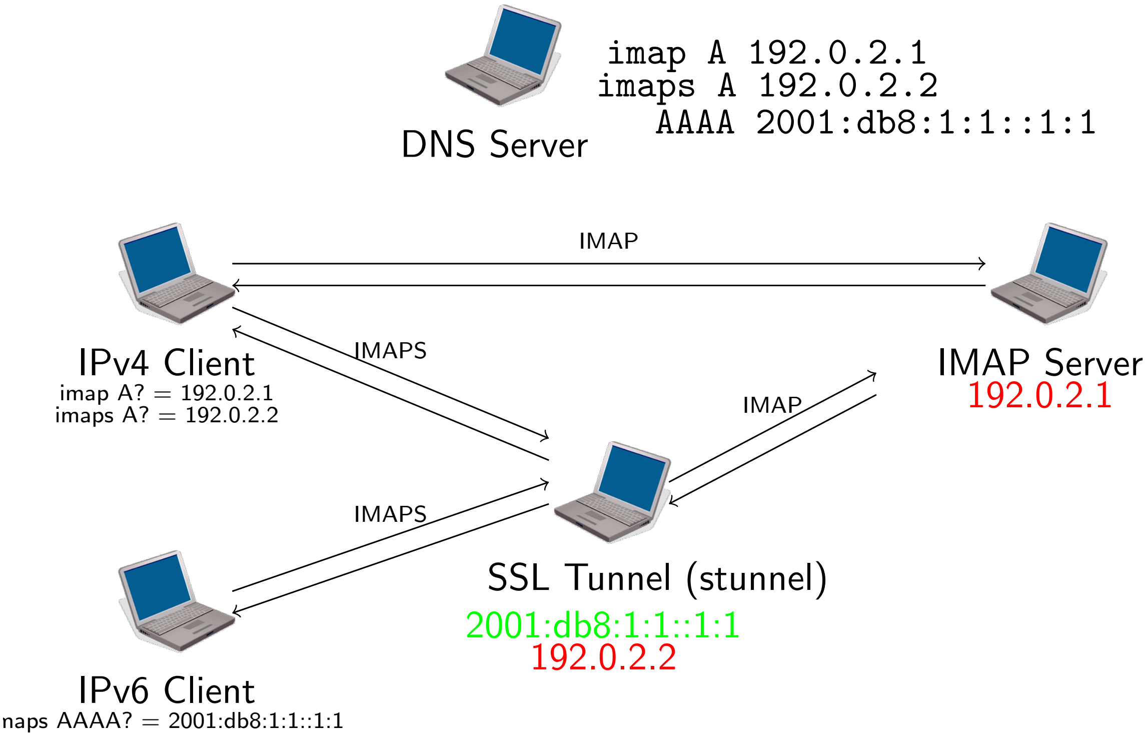
HTTP Server
192.0.2.1

HTTP Proxy (Apache)
2001:db8:1:1::1:1
192.0.2.2

IPv6 Client
vww AAAA? = 2001:db8:1:1::1:1

SSL Tunnel

How to enable IPv6 access to a production Mail server



```
 imap A 192.0.2.1
imaps A 192.0.2.2
  AAAA 2001:db8:1:1::1:1
```

DNS Server

IMAP

IMAPS

IMAP

IMAPS

IPv4 Client
imap A? = 192.0.2.1
imaps A? = 192.0.2.2

IMAP Server
192.0.2.1

SSL Tunnel (stunnel)
2001:db8:1:1::1:1
192.0.2.2

IPv6 Client
naps AAAA? = 2001:db8:1:1::1:1

Monitor IPv6 usage

Monitoring IPv6 is important for

- See impact of IPv6 deployement
- Ensure same Quality of Service in IPv4 an IPv6

Tools

- Traffic: MRTG/Cacti, Netflow v9. . .
- Services: Nagios, Zabbix. . .

Dual-Stack requires dual check !

Need to check service reachability BOTH in IPv4 AND in IPv6

Integration

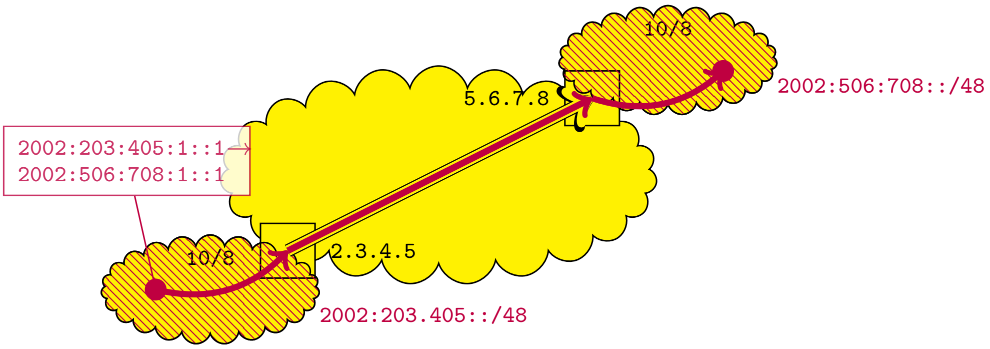
Home network and SOHO

Home Network

- Must (should) be transparent for the end-users
- Last Mile is not currently v6fied
- Wait .... or used Tunnel Brokers
  - DO NOT USE TEREDO OR 6to4
- homenet IETF working group specifies home network behavior for IPv6
  - Today: star topology around single CPE
  - Tomorrow: Mesh network and multi-homing
    - Internet of things
    - smart grid
    - . . .

6to4

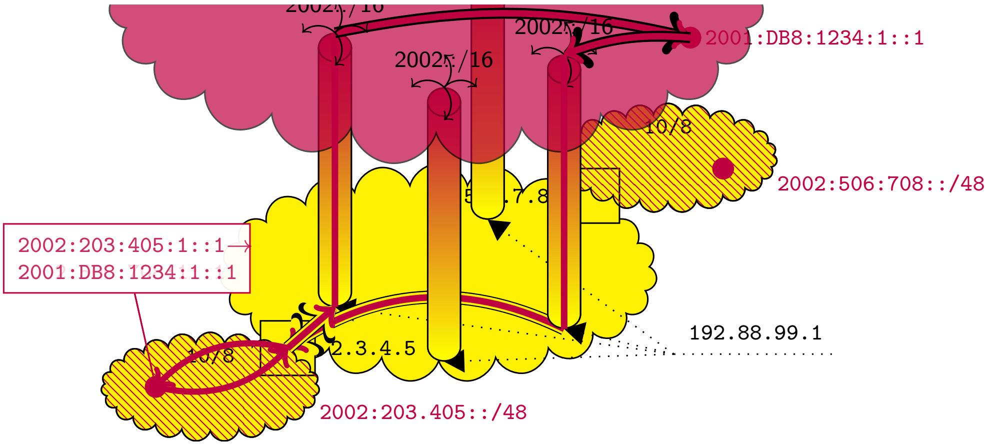
- based on the magic formula 16+32=48
  - 2002::/16 + IPv4 address



- Cannot cross NAT (need to know public address)
- Bad performances.

6to4

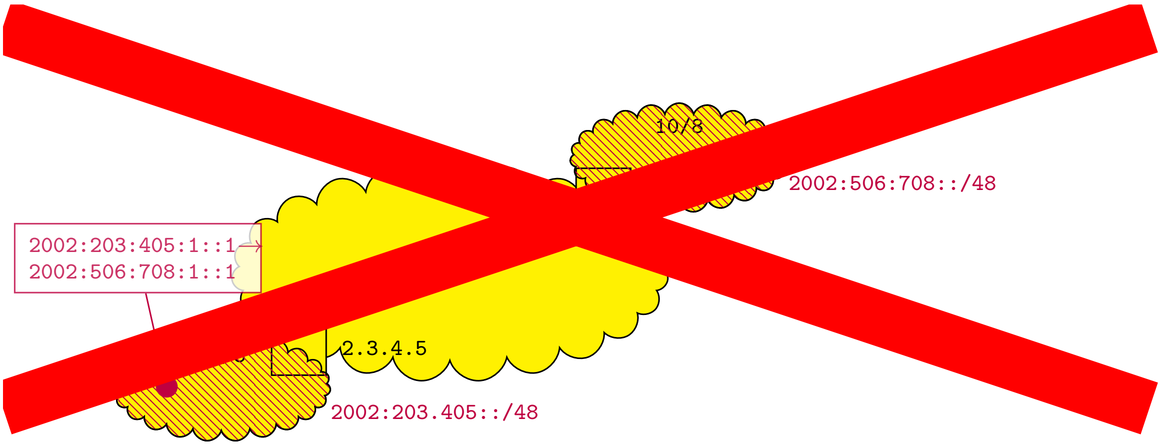
- based on the magic formula 16+32=48
  - 2002::/16 + IPv4 address



- Cannot cross NAT (need to know public address)
- Bad performances.

6to4

- based on the magic formula 16+32=48
  - 2002::/16 + IPv4 address

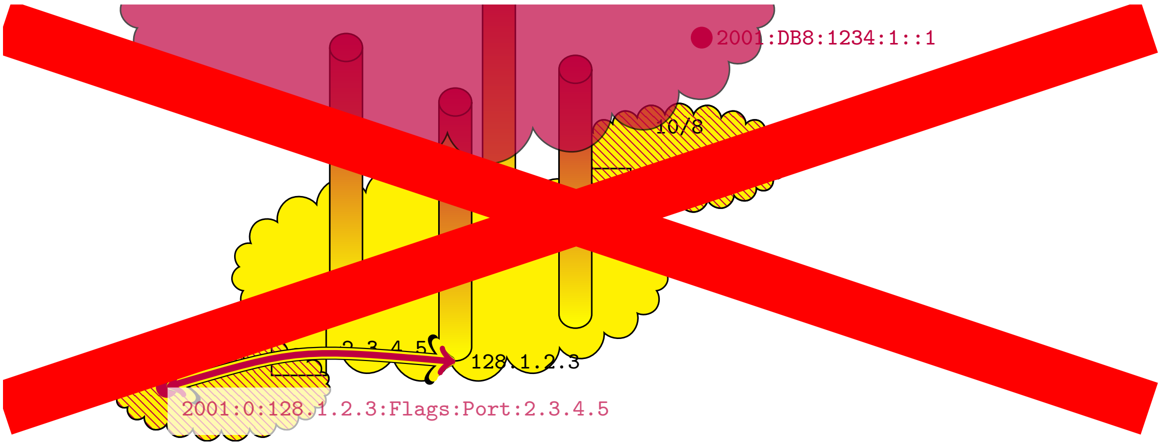


- Cannot cross NAT (need to know public address)
- Bad performances.

TEREDO

Integration
Why IPv6 Integration ?
6 generic scenarios
Tools overview
Scenarios
Backbone operator
Internet Access Provider
3G/LTE
Enterprise
Home network and SOHO
Programming IPv6 Applications

- Based on NAT Traversal protocol
  - 2001::/32 allocated to this mechanism.



Performances?

Integration
Why IPv6 Integration ?
6 generic scenarios
Tools overview
Scenarios
Backbone operator
Internet Access Provider
3G/LTE
Enterprise
Home network and SOHO
Programming IPv6 Applications

- If performances with 6to4 and TEREDO are worst than with IPv4
- What happens if a site decides to activate dual stack on its servers ?
  - Customers will run away
- if IPv6 is dead
  - client starts will IPv6 and then after a long timeout tries IPv4
  - bad performances
- Happy Eyes Ball: try IPv4 and IPv6 simultaneously
- Test the same day IPv6 on main sites
  - Customer will not run away

Performances?

- the 6/8/11: v6Day
  - Good news: nobody notice it
  - 0.3% of IPv6 traffic
- Conclusion: Activating IPv6 do not create troubles
- 6/6/12: IPv6 activated on main sites (google, yahoo, facebook, akamai,. . .)
  - Potentially 50% of Internet traffic
  - in reality less since access network is missing

IPv6 socket API in C, C++

Socket API

Integration

Programming IPv6 Applications

CC++ API

JAVA API

- Socket Unix API has been extended to IPv6
- New protocol and address family PF_INET6 and AF_INET6
- New structures :
  - `in6_addr`
  - `sockaddr_in6`
  - `sockaddr_storage`
- New functions for names to addresses conversion

Reference

RFC 2553 & Posix 1003.1g

Structure for sockets

Integration

Programming IPv6 Applications

CC++ API

JAVA API

Structure in C, C++

```
struct sockaddr_in6 {
  uint8_t          sin6_len;       /* structure length
  sa_family_t      sin6_family;    /* AF_INET6
  in_port_t        sin6_port;      /* transport layer port
  uint32_t         sin6_flowinfo;  /* IPv6 traffic class & flow info
  struct in6_addr  sin6_addr;      /* IPv6 address
  uint32_t         sin6_scope_id;  /* set of interfaces for a scope
};
```

- Similar to sockaddr_in for IPv4
- New fields for scope and flow label

`sizeof(sockaddr_in6) > sizeof(sosckaddr_in)`

- sockaddr_in6 can not be stored in `struct sockaddr`
- Programs have to be modified to be AF-independent !

Managing Sockets in C, C++

Managing sockets

- Creation : Same as in IPv4
  - `int s = socket(PF_INET6, SOCK_STREAM, 0);`
- Other functions are not modified
  - `bind, connect, listen, accept, send*, recv*, getpeername, getsockname`
- New functions to manage options
  - `getsockopt, setsockopt`

Sockets and address families

Integration
Programming IPv6 Applications
CC++ API
JAVA API

2 options for applications :

- Only use PF_INET6 socket
  - On a IPv4 networks, use IPv4-mapped IPv6 addresses
  - Problem: when IPv6 stack is not available ...
- Use one PF_INET socket and one PF_INET6 socket
  - Client knows which socket to open with `getaddrinfo`
  - Server should wait for packets on both sockets

Examples found with `netstat -taun` (MacOSX)

Proto Rec Send Local Foreign State
tcp46 0 0 *.80 *.* LISTEN ← Apache server uses first option
...
tcp4 0 0 *.22 *.* LISTEN ← SSH server uses second option
tcp6 0 0 *.22 *.* LISTEN ←

Example : Client connection

Integration
Programming IPv6 Applications
CC++ API
JAVA API

```
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netdb.h>
int open_conn(const char *host) {
    int sock = -1, ecode;
    struct addrinfo *res, *r, hints = {
        0, PF_UNSPEC, SOCK_STREAM, 0};

    if ((ecode = getaddrinfo(host, "daytime", &hints, &res)))
        errx(1, "getaddrinfo: %s", gai_strerror(ecode));
    for (r = res; r && sock < 0; r = res->ai_next)
        if ((sock = socket(res->ai_family, res->ai_socktype, res->ai_protocol)) < 0 ||
            connect(sock, res->ai_addr, res->ai_addrlen))
            sock = -1;
    freeaddrinfo(res);
    return sock;
}
```

Example : Server socket

```
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netdb.h>
int open_serv(const char *serv) {
    int sock, ecode;
    struct addrinfo *res, hints = {
        AI_PASSIVE, PF_UNSPEC, SOCK_STREAM, 0};

    if ((ecode = getaddrinfo(NULL, serv, &hints, &res))
        errx(1, "getaddrinfo: %s", gai_strerror(ecode));
    if ((sock = socket(res->ai_family, res->ai_socktype,res->ai_protocol)) < 0) ||
         bind(sock, res->ai_addr, res->ai_addrlen) ||
        listen(sock, 1))
        err(1, "socket");
    freeaddrinfo(res);
    return sock;
}
```

Example : Server connection

```
main() {
    int sock = open_serv("1000");

    for(;;) {
        struct sockaddr_storage from;
        int s, len = sizeof from;
        char name[NI_MAXHOST];

        if ((s = accept (sock, (struct sockaddr*)&from, &len) < 0)
            err(1, "accept");

        if (getnameinfo((struct sockaddr*)&from, &len, name,
                sizeof name, NULL, 0, NI_NUMERICHOST))
            name[0] = 0;
        printf("connexion %s\n", name);
        /* utiliser socket s ? */
        close (s);
}
```

Rules to anticipate integration of IPv6 protocol

Generic structure for sockets

- Programs should use `struct sockaddr_storage` to be AF-independent
- Cast depending of AF when needed

Socket containers

```
struct sockaddr_storage ss;
foo((struct sockaddr *)&ss);    // AF independent function

void foo(struct sockaddr *s) {
    // If we need IPv4 socket
    struct sockaddr_in *sin = (struct sockaddr_in *) s;
    // If we need IPv6 socket
    struct sockaddr_in6 *sin6 = (struct sockaddr_in6 *) s;
}
```

Address manipulation : getaddrinfo()

Integration
Programming IPv6 Applications
CC++ API
JAVA API

getaddrinfo() Prototype

```
int getaddrinfo(const char *nodename,
                const char *servname,
                const struct addrinfo *hints,
                struct addrinfo **res);
```

- Generic function for name resolution, AF-independent
- Replace function `gethostbyname`
- `servname`: String for protocol name ("http") or port number ("80")
- `hints`: Refine request (IPv4 only, IPv6 only, IPv4/IPv6)
- May return more than one result !

Address manipulation : getnameinfo()

Integration
Programming IPv6 Applications
CC++ API
JAVA API

getnameinfo() Prototype

```
int getnameinfo(const struct sockaddr *sa,
                socklen_t salen,
                char *host,
                socklen_t hostlen,
                char *serv, socklen_t servlen,
                int flags);
```

- Generic function for reverse resolution, AF-independent
- Replace function `gethostbyaddr`

Macros

Macros to test nature of address:

- `IN6_IS_ADDR_UNSPECIFIED (struct in6_addr *);`
- `IN6_IS_ADDR_LOOPBACK (struct in6_addr *);`
- `IN6_IS_ADDR_MULTICAST (struct in6_addr *);`
- `IN6_IS_ADDR_LINKLOCAL (struct in6_addr *);`

Macros to test address equality :

- `IN6_ARE_ADDR_EQUAL (struct in6_addr *, struct in6_addr *);`

Migrate existing applications

Porting applications to IPv6 (in a nutshell)

1: Replace IPv4-only structures and functions with AF-independent version

Generic Structure & Functions

`hostent` → `addrinfo`
`sockaddr_in` → `sockaddr_storage`
`gethostbyname` → `getaddrinfo`
`gethostbyaddr` → `getnameinfo`

2: Look for particular usage of IP address structure `in_addr`

- Applications sometimes use IP addresses as host identifier
- This should be made AF-independent

Porting applications to IPv6 (in a nutshell)

3: Choose a strategy when opening socket (one or two sockets ?)

4: Consider one host may have more than one address !

- With `getaddrinfo` you may have one IPv4 and several IPv6 addresses for one host
- To be also considered when using address as host identifier

5: Beware of textual representation of IP addresses

Beware

http://[2001:660:7301:1::1]
scp foo.bar [2001:660:7301:1::1]:/tmp

IPv6 JAVA API

IPv6 Support in Java

- Java support IPv6 since JDK 1.2, extended with JDK 1.4
- Extension have been made for class InetAddress
- Inheritance and polymorphism ensures relative transparency for version of manipulated addresses

Inet6Address

New subclass of `InetAddress` (with `Inet4Address`)

- Class for instanciate IPv6 addresses
- Methods for checking address scope :
  - `isIPv4CompatibleAddress` (for IPv4-mapped addresses)
  - `isLinkLocalAddress`
  - `isMulticastAddress`

InetAddress

`InetAddress` objects may be either IPv4 or IPv6 address

`InetAddress` class extended for DNS resolution

- Method `getByName` returns only IPv4 name resolution
- New method `getAllByName` returns all possible name resolutions (IPv4 and IPv6)
- Reverse resolution unchanged

Changes for IPv6 support

Name resolution using `getByName` should be changed to use `getAllByName` and uses the returned array of addresses

Socket API

- Socket API is based on super-class InetAddress → no major change
- By choosing binding address, change protocol enabled for socket
  - IPv4 binding address → Socket listening for IPv4
  - IPv6 binding address → Socket listening for IPv4 and IPv6

Consequences

- Integration of IPv6 is harmless for IPv4 operations
- IPv6 will be used when correspondant address is IPv6