Zurich Research Laboratory

Applications of IPv6

Jeroen Massar
jma@zurich.ibm.com

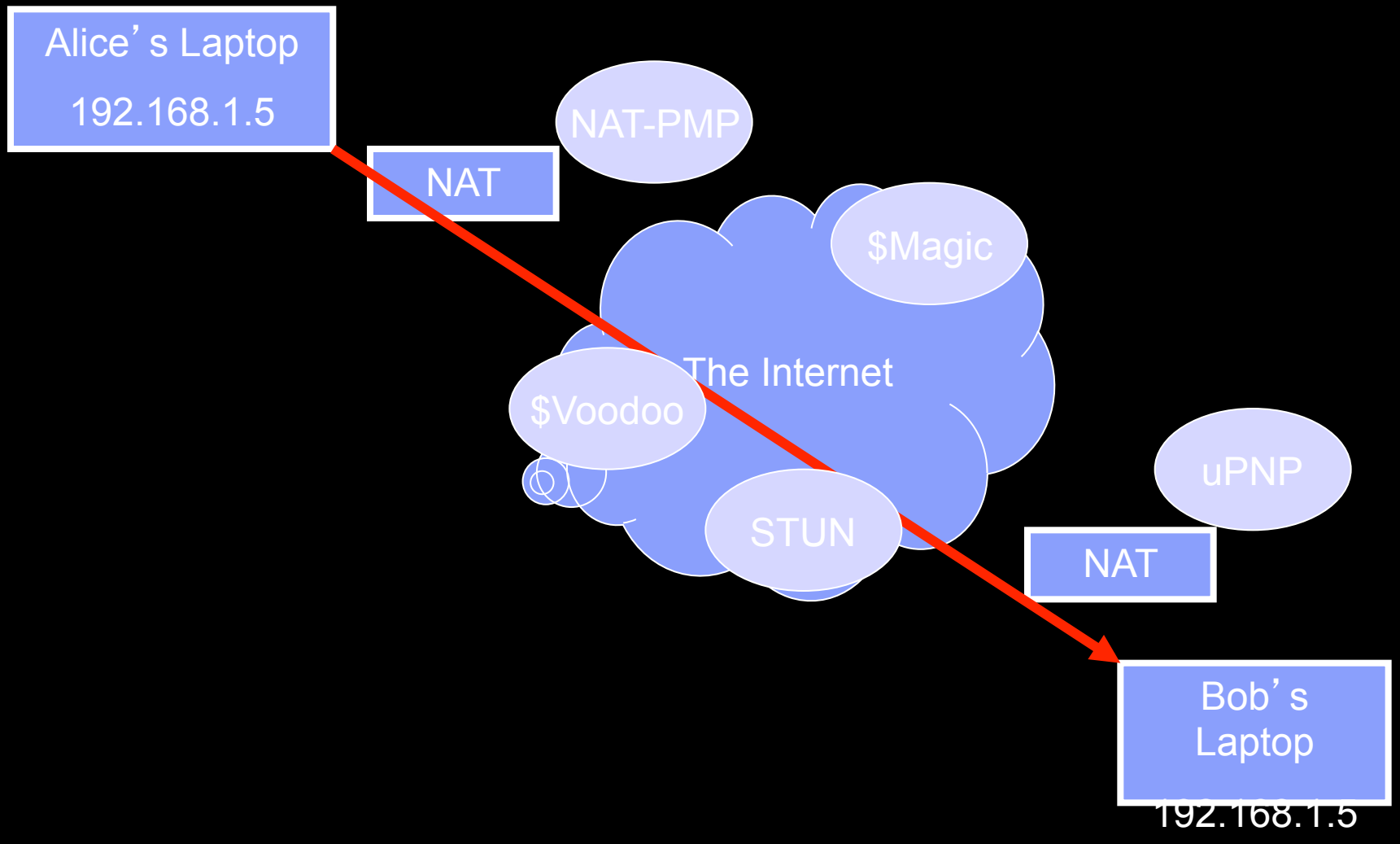
2nd German IPv6 Summit
Hasso-Plattner-Institute, Potsdam, Germany
2009-05-15 | Jeroen Massar <jma@zurich.ibm.com>

Who is Jeroen?

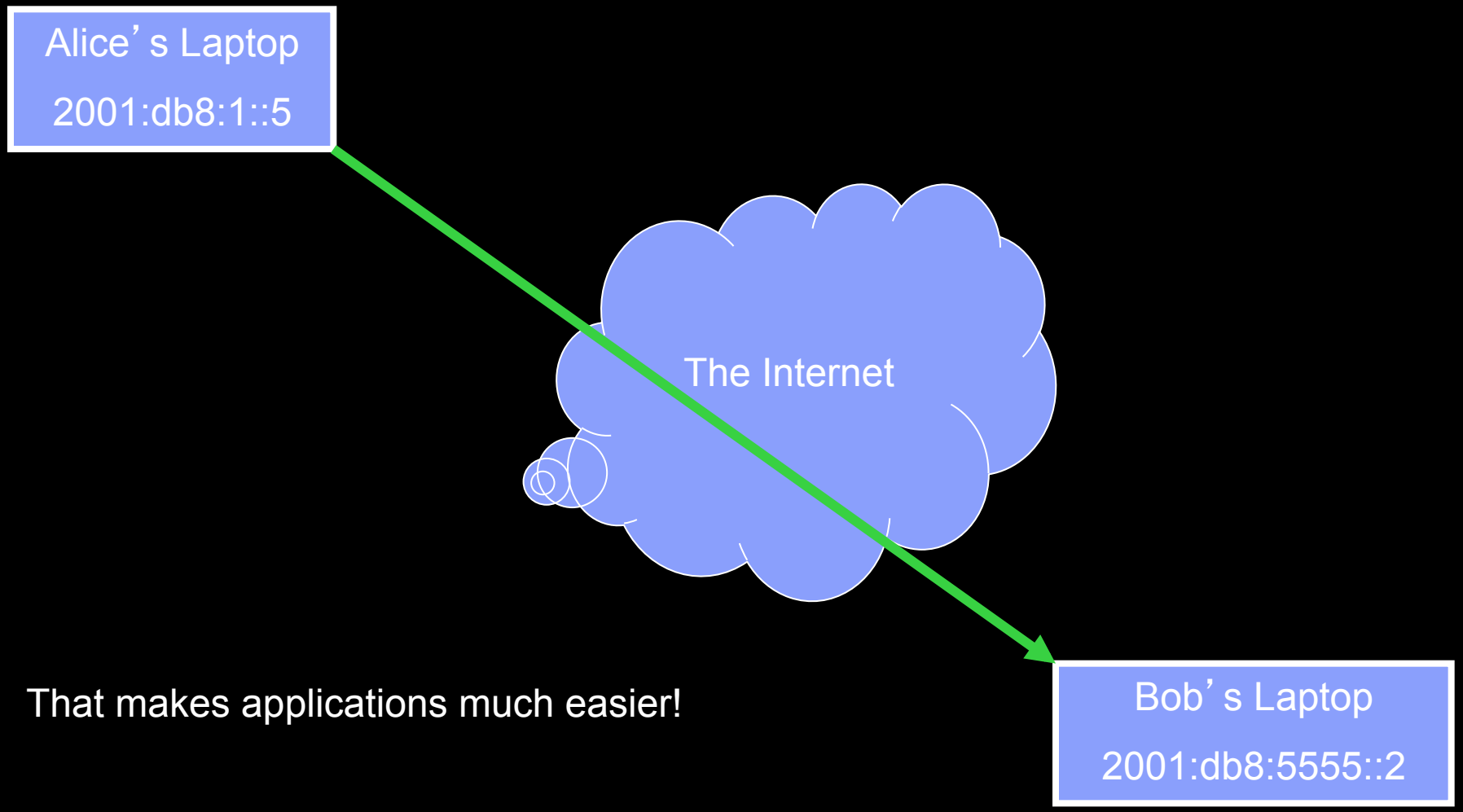
- Working at IBM Zurich Research Laboratory which is located in Rüschlikon, doing IPv6 related projects and assignments and AURORA (http://www.zurich.ibm.com/aurora/) a high performance network analyzer.
- “Small” spare-time hobby project since 2001 named SixXS (http://www.sixxs.net/) which provides free IPv6 connectivity to users worldwide along with GRH for IPv6 routing monitoring.
- Contributor to IETF & IRTF on various IPv6 and routing related subjects. Made amongst others PuTTY compatible.

  Other details: http://unfix.org/~jeroen/ http://www.zurich.ibm.com/~jma/

IPv4 Today



But with IPv6…



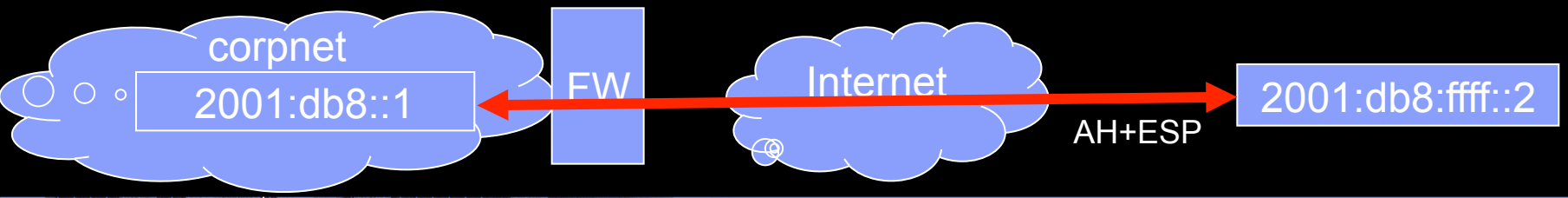
That makes applications much easier!

or with Microsoft Direct Access-style using IPSEC

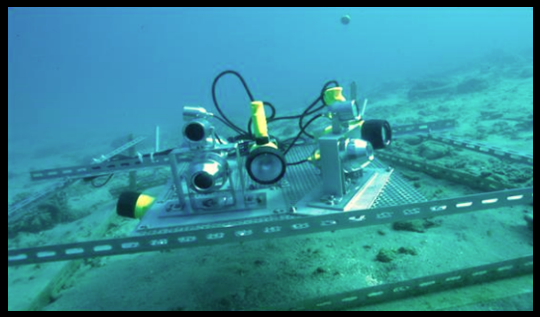
- How to do ‘firewalling’ without having to configure prefixes.
- You have your host anywhere, just have IPv6, be that native, proto-41, Teredo, 6to4 or through a Tunnel Broker, could be remote or in the office.
- Sign every packet you sent/receive from your client to the server with an IPSEC-AH.
- Firewall in the middle allows any correctly signed and validated packet, of course only accepting keys that it knows.
- Use IPSEC-ESP to secure packets so that nobody can snoop them.
- Key distribution happens using Active Directory.

- The Direct Access tool allows one to directly access any resources one has at the workplace from any other location on the (inter-)net.

- Same method can be applied to any kind of connection of course.



IPv6 Toys: Home automation, fridges, sensors, etc

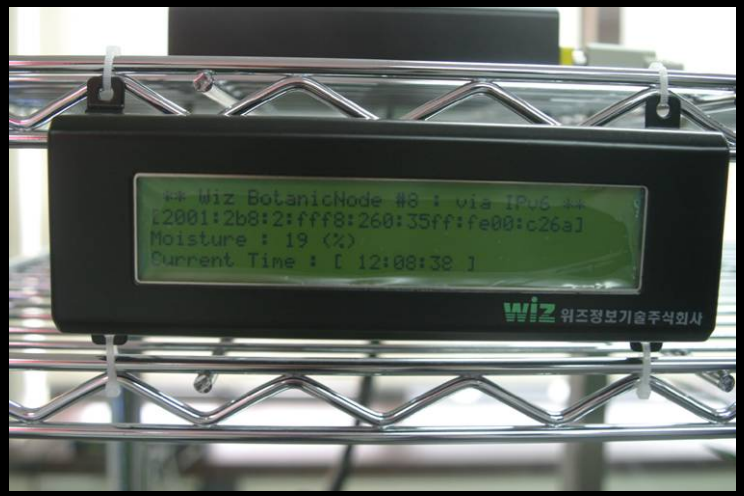


For more:

google(IPv6 toys)

google(IPv6 cool)

IPv6 Toys: $ telnet plant

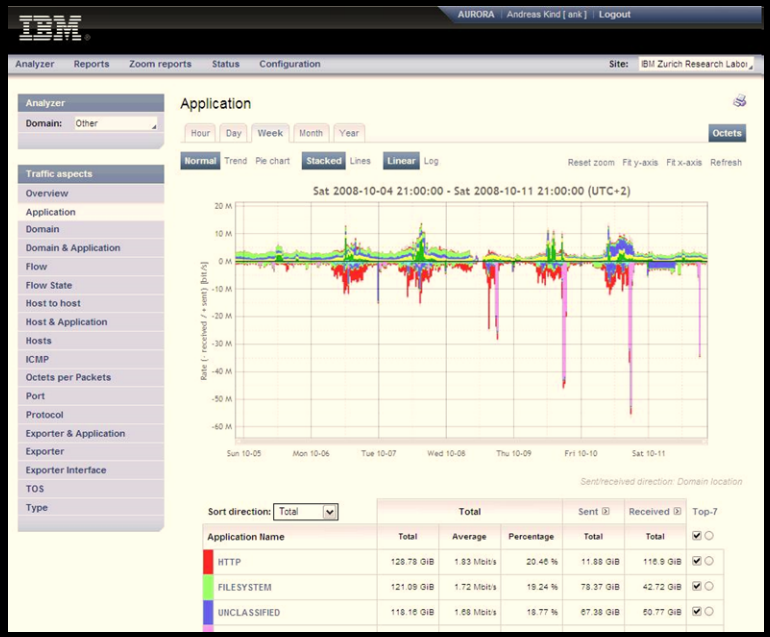


Finding IPv4 applications

- Monitor your network with NetFlow / sFlow and similar protocols
  We have a tool called AURORA for this purpose which does dependency mapping

- Programs using IPv4 might do so because:
  - No IPv6 support on the client
  - The client code doesn't support IPv6
  - The server doesn't have an AAAA record or other IPv6 referral
  - No IPv6 support on the server
  - The server code doesn't support IPv6
  - The Operating System is configured to prefer IPv4 over IPv6 (RFC3484)

  … networking issues etc etc.…



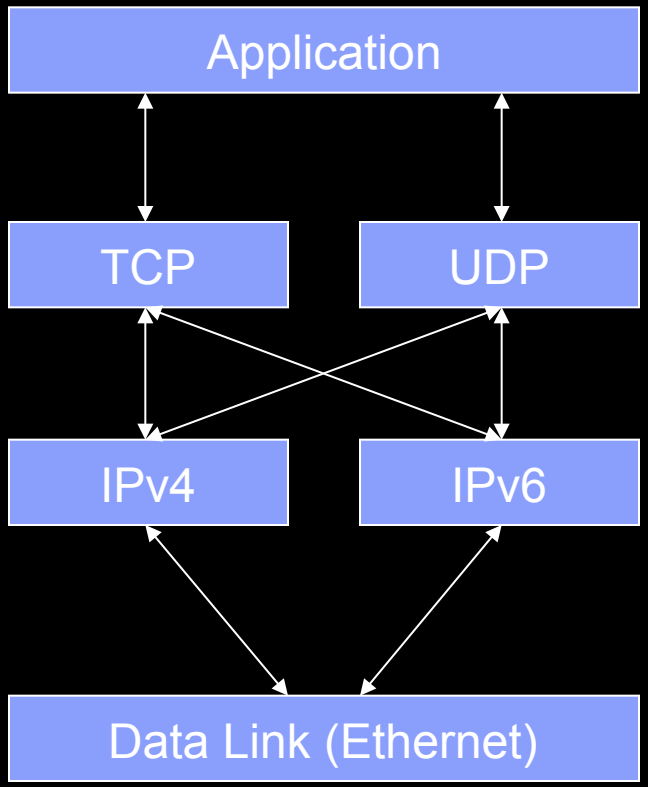
http://www.zurich.ibm.com/aurora/

Note that Dual-Stack is a GOOD thing

Dual stack operation

- Applications can use both IPv4 and IPv6 at the same time, protocols can coexist.
- DNS can contain both IPv4 and IPv6 addresses.
- IPv6 applications can use IPv4-mapped addresses. Though this only is available when the kernel has a mixed IPv4/IPv6 stack. Windows IPv6 implementation doesn't have this for instance.

- Best option: Address Family independency



Address Family (AF) independency

New functions from RFC2553 and RFC2292:

- getaddrinfo()

  Getting the addresses belong to a textual identifier

  Replaces:

  - gethostbyname()
  - getservbyname()
  - inet_pton()

- getnameinfo()

  Getting a textual representation of an address

  Replaces:

  - getservbyaddr()
  - getservbyport()
  - inet_pton()

Porting considerations

- Change socket functions
- Adjust logging function so they can handle larger IP address (Don't forget to store both the hostname and the address as changing reverses is as easy as getting a new IP, having two tracking points is better)
- Increase all data member that stores IP addresses in program and in databases/configuration files
- Adjust keyboard and display interface function so they can handle larger IP addresses.
- RFC2732 states that IPv6 addresses in URIs should be delimited by square brackets [ ] which solves the problem where applications use the colon (:) to distinguish the port from the address (eg [2001:db8::1]:80).
- getaddrinfo() returns 0 on succes, !0 on failure unlike most other unix/posix calls.

Porting applications / coding new ones

- google(eva ipv6) or http://gsyc.escet.urjc.es/~eva/IPv6-web/ipv6.html
- Contains “Porting applications to IPv6 HowTo“ with great explanation and example code.
- Also serves of course as a rather good example for new programs.

- Implementing AF-independent application document by Jun-ichiro ‘itojun’ Itoh (http://www.kame.net/newsletter/19980604/)
- The document from the master IPv6 Samurai himself.

Server code example

Daemons should listen on all possible combinations, as PF_UNSPEC, mostly 2 maximum (IPv4+IPv6)

```
int sockets[10];
int makelisten(char *server, char *service)
{
    int i = 0;
    struct addrinfo hints;
    struct addrinfo *res;
    memset(&hints, 0, sizeof(hints)); /* set-up hints structure */
    hints.ai_family = PF_UNSPEC;
    hints.ai_flags = AI_PASSIVE;
    hints.ai_socktype = SOCK_STREAM;
    if ((error = getaddrinfo(server, service, &hints, &res)))
    {
            perror(gai_strerror(error));
            return -1;
    }
    while (res)
    {
            sockets[i] = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
            if (sockets[i] == -1) continue;
            if (bind(sockets[i], , res->ai_addr, res->ai_addrlen) == 0 &&
                listen(sockets[i]) == 0) { i++; continue; }
            close(sockets[i]); sockets[i] = -1;
    }

}

int socket = makelisten(NULL, "80");
```

Client code example

Client side program should try to connect to all resolved addresses,

```
int makeconnect(char *server, char *service)
{
    struct addrinfo hints;
    struct addrinfo *res;
    memset(&hints, 0, sizeof(hints)); /* set-up hints structure */
    hints.ai_family = PF_UNSPEC;
    hints.ai_socktype = SOCK_STREAM;
    if ((error = getaddrinfo(server, service, &hints, &res)))
    {
            perror(gai_strerror(error));
            return -1;
    }
    // Loop through all the possiblities
    while (res)
    {
            sockfd = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
            if (sockfd == -1) continue;
            if (connect(sockfd, res->ai_addr, res->ai_addrlen) == 0) return sockfd;
            close(sockfd);
    }
    return -1;
}

int socket = makeconnection("www.zurich.ibm.com", "80");
```

RFC3484 – Address Ordering

- Orders addresses:
  - IPv6 native
  - IPv4 native
  - 6to4
  - Teredo
- Configurable
  - Linux: /etc/gai.conf
  - Solaris: ipaddrsel
  - FreeBSD: ip6addrctl
  - Windows: netsh

Use a socket per AF

```
tcp6       0      0 :::993                  :::*                    LISTEN      -
```

Does that listen on IPv6 only? -> Not on Linux, there it is IPv4 and IPv6, applications need to know this as they will see incoming connections on IPv4 as ::ffff:192.0.2.42

```
tcp        0      0 0.0.0.0:993             0.0.0.0:*               LISTEN      -
tcp6       0      0 :::993                  :::*                    LISTEN      -
```

Security implications!

- More diverse devices connected, thus possibly also more vulnerabilities.
- Firewalls should be enabled per default for all incoming connections.
- Scanning of address space is not feasible
  - /64 per link
  - /48 or /56 per 'endsite'
- Currently only BSD has a stateful firewall, Linux since 2.6.[5|6]-USAGI. Cisco PIX has it too, but loadbalancing broken.

Current IPv6 Application / Deployment Issues

- IPv6 is not everywhere yet, and IPv4 hosts are behind NAT
  - Tunnel: 6to4 / Teredo / AYIYA / static tunnels
- DNS issues in NAT-machines (aka home user CPE ‘router’)

  These hosts drop non A/TXT/NS record queries, thus especially AAAA which causes “IPv6 to make the Internet slow”.
  - Get a fixed firmware (though sometimes
  - Use the DNS server of the upstream directly
- Broken connectivity
  - Routing issues
  - pMTU issues
  - Firewalls

The End

Comments or questions?

2nd German IPv6 Summit
Hasso-Plattner-Institute, Potsdam, Germany
2009-05-15 | Jeroen Massar <jma@zurich.ibm.com>

References

- IETF – The Internet Engineering Task Force
  http://www.ietf.org
- IRTF – The Internet Research Task Force
  http://www.irtf.org
- IVI
  http://www.ivi2.org
- Patrick Fältström's IETF photo site
- SixXS - IPv6 Deployment & Tunnel Broker
  http://www.sixxs.net