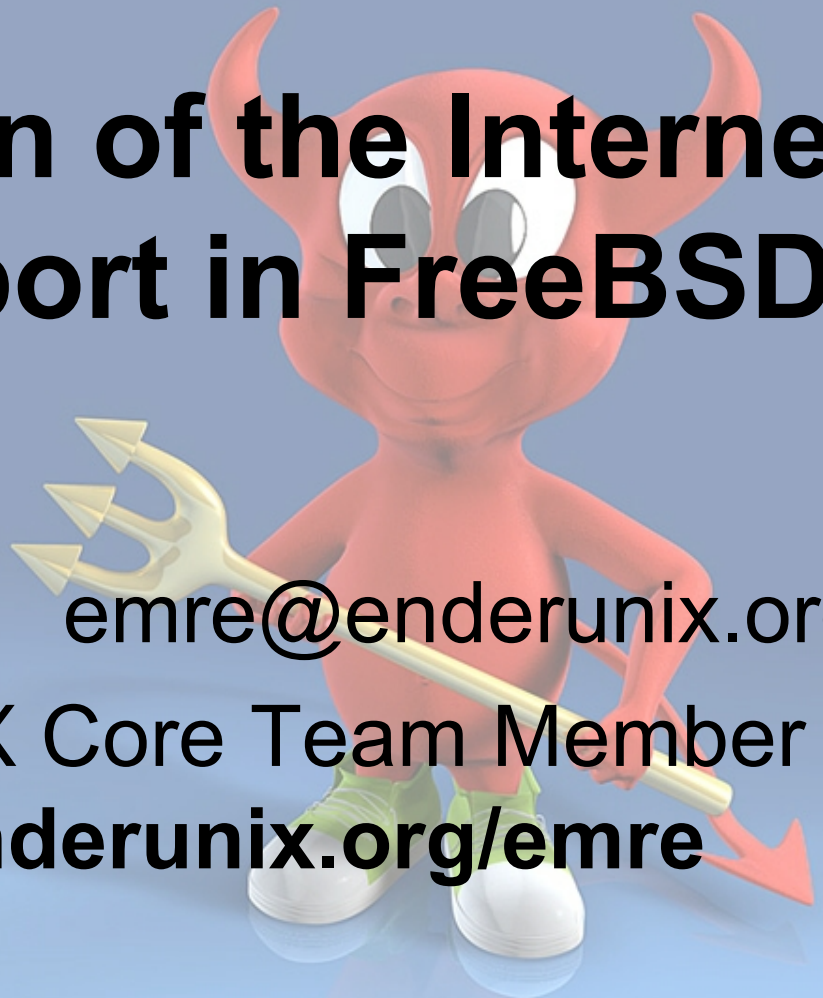
Resurection of the Internet: IPv6 support in FreeBSD



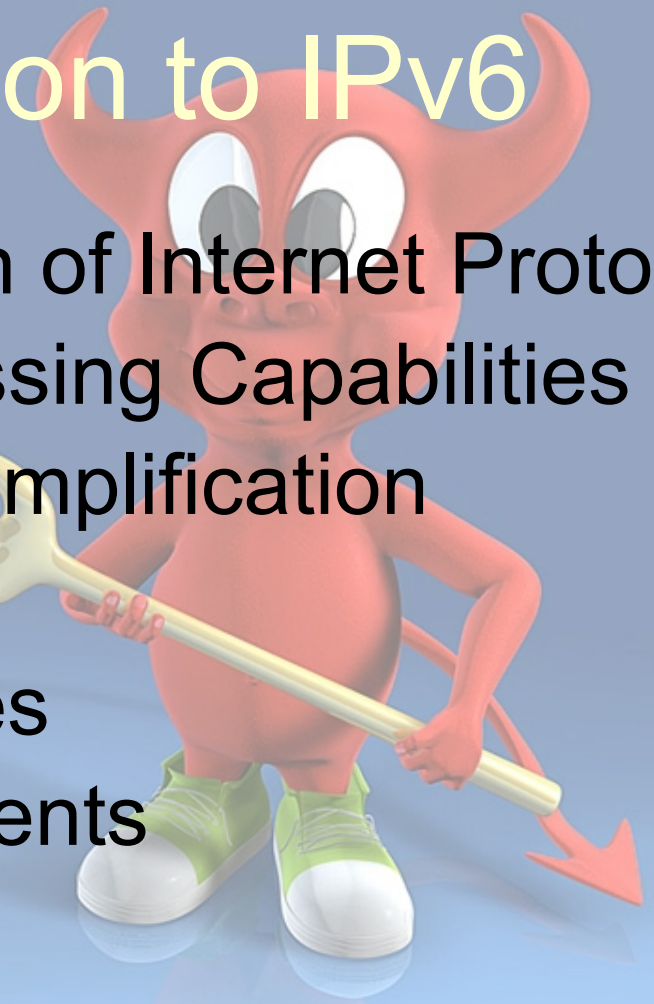
Bâkır EMRE emre@enderunix.org

EnderUNIX Core Team Member

www.enderunix.org/emre

Introduction to IPv6

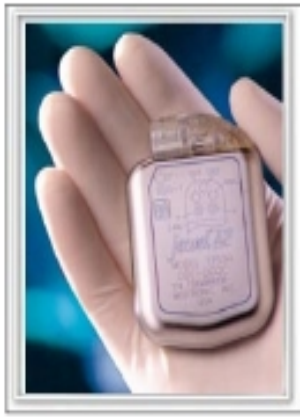
- IPv6: New version of Internet Protocol
- Expanded Addressing Capabilities
- Header Format Simplification
- Replaces IPv4
- Many new features
- Critical improvements



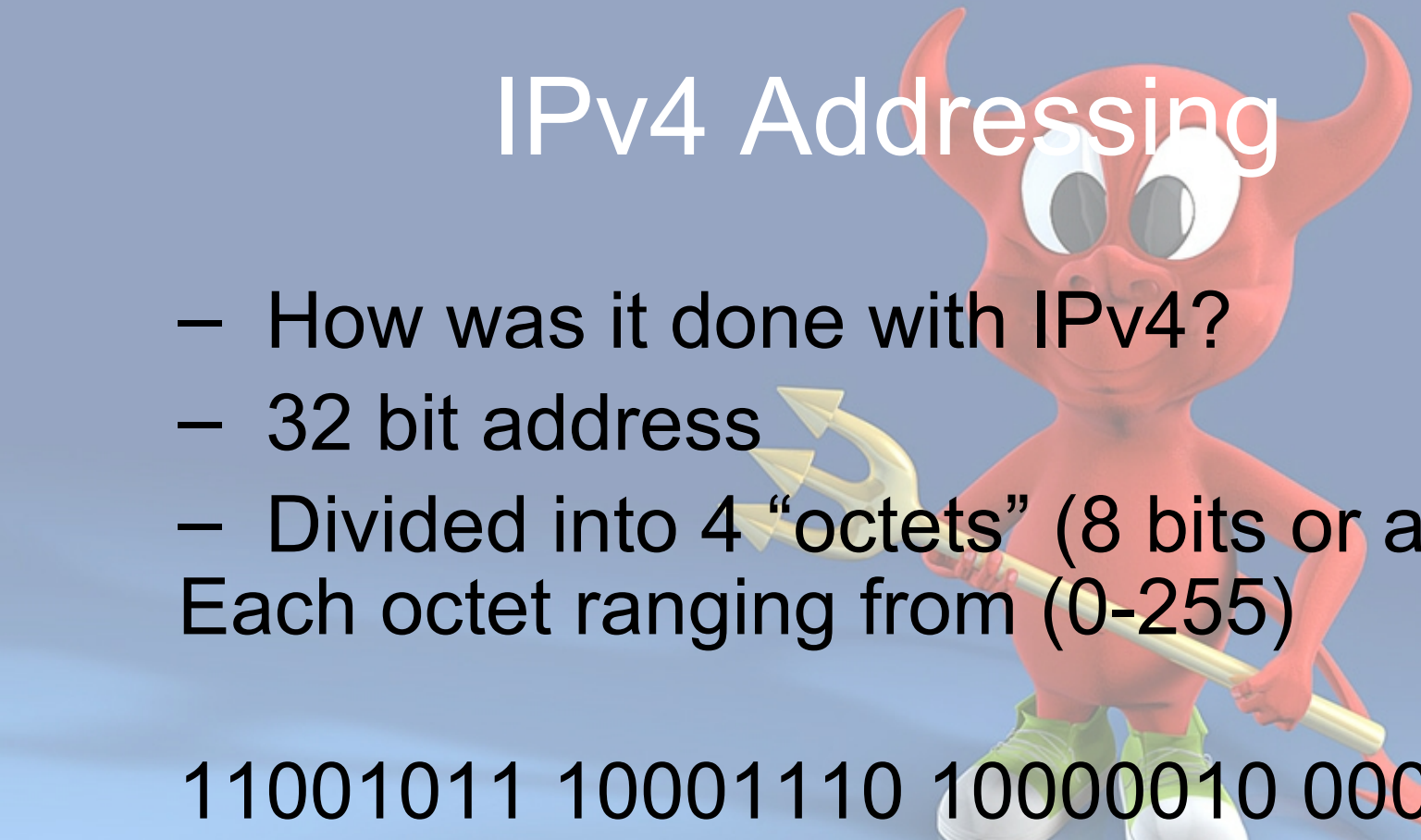
Why we need IPv6

- Exhaustion of available IPv4 address space
- Autoconfiguration
- plug-and-play support
- Embeded Security
- Better network performance
- Built-in mobility


msn radio
Rocking Alternative Hits
radioplus



IPv4 Addressing



- How was it done with IPv4?
- 32 bit address
- Divided into 4 “octets” (8 bits or a byte)

Each octet ranging from (0-255)

11001011 10001110 10000010 00000101

203 . 142 . 130 . 5

IPv6 Addressing

- Represented in Hex (every 4 bits)

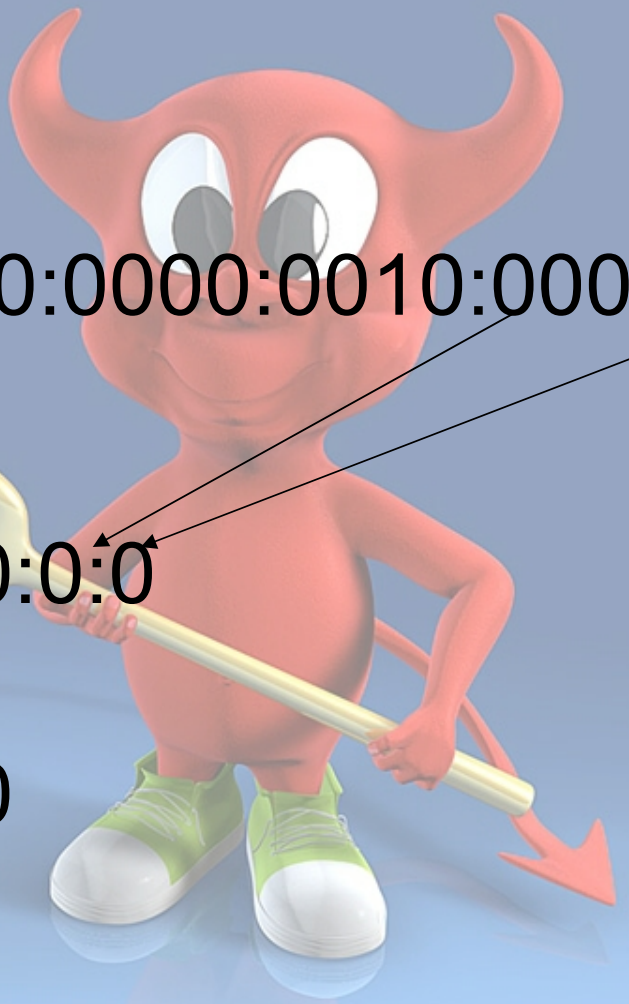
0010 0000 0000 0001 0000 1101 1011
1000 0000 0000 0000 0010 0000 0000
0000 0000 0000 0000 0000 0000 0000
0000 0001 0000 0000 0000 0000 0000
0000 0000 0000 0000

2001:0DB8:0002:0000:0000:0010:0000:0000

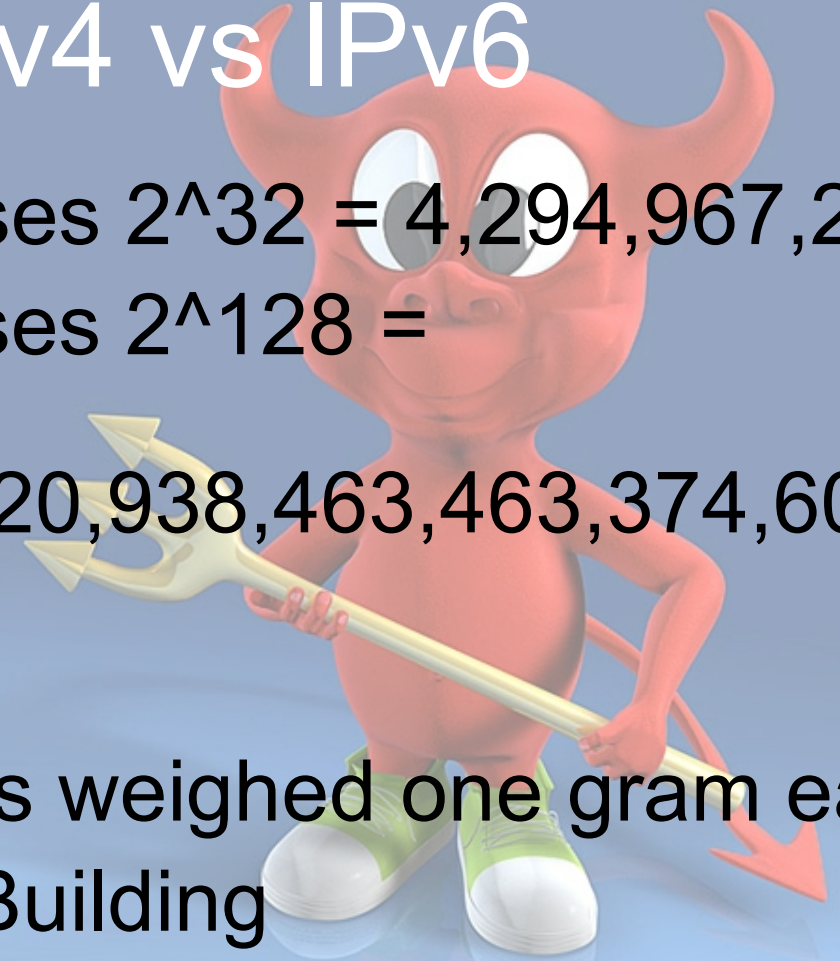
2001:0DB8:0002:0000:0000:0010:0000:000

– 2001:0DB8:2:0:0:10:0:0

–2001:0DB8:2::10:0:0



IPv4 vs IPv6



- IPv4 addresses 2^32 = 4,294,967,296
- IPv6 addresses 2^128 =

340,282,366,920,938,463,463,374,607,431,768,211,456

If IP addresses weighed one gram each

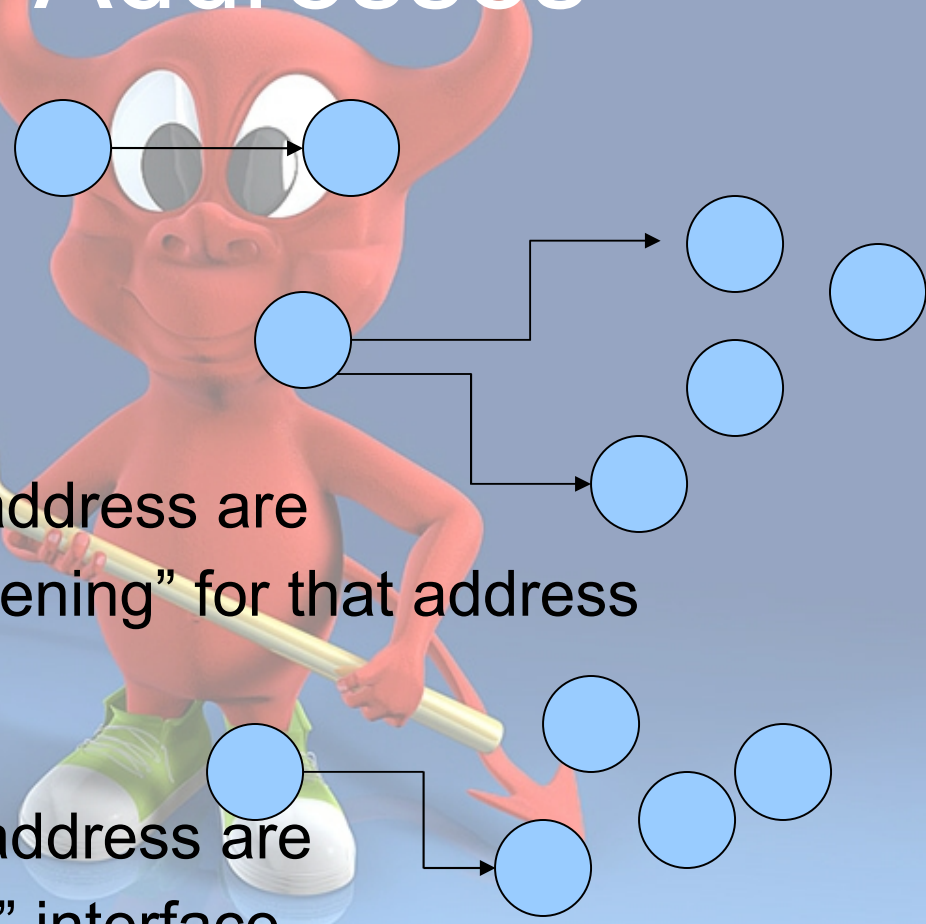
- IPv4 < One Building
- IPv6 > 56 billion Earths

IPv6 Type of Addresses

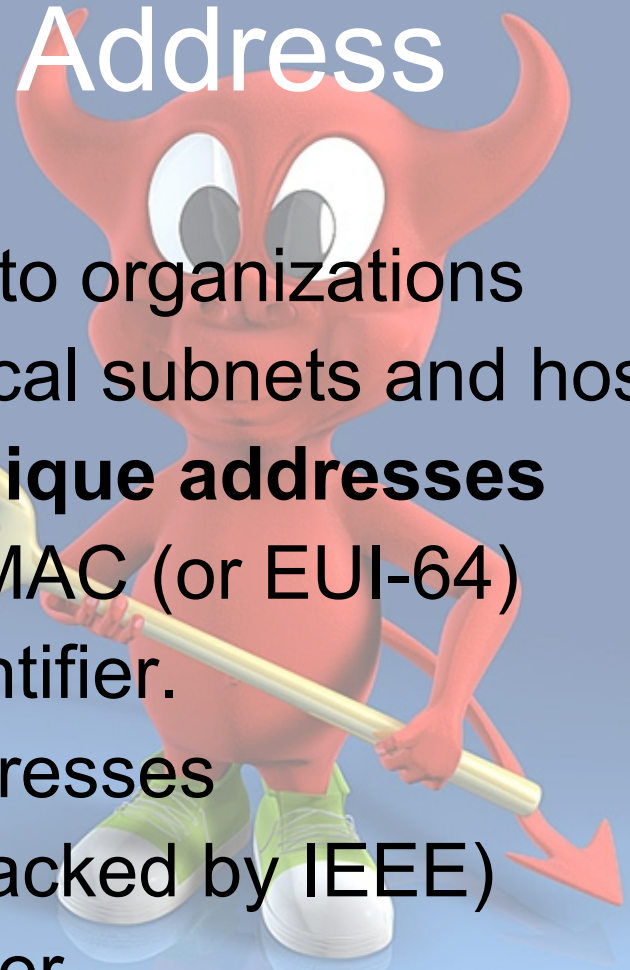
- **Unicast**
  - Identifies a single interface

- **Multicast**
  - Packets sent to a multicast address are delivered to all interfaces “listening” for that address

- **Anycast**
  - Packets sent to an anycast address are delivered to a single “nearest” interface



Unicast Address



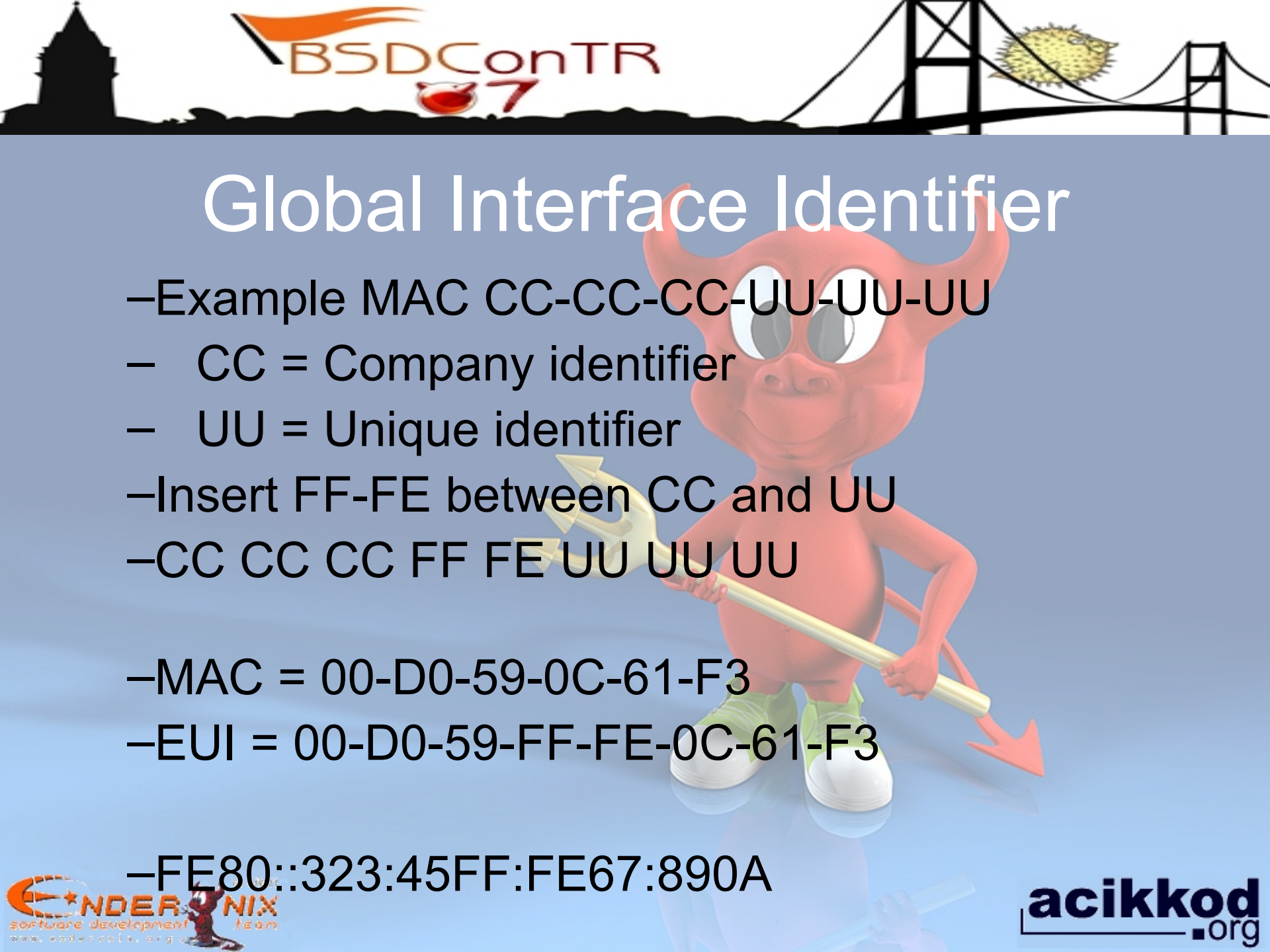
- ISPs assign prefixes to organizations
- Organizations add local subnets and host IDs
- Result is **globally-unique addresses**
- Ethernet builds on MAC (or EUI-64)
- Extended Unique Identifier.
- Will replace MAC addresses
- 24-bit company_id (tracked by IEEE)
- 40-bit “unique” identifier

Global Interface Identifier

- Example MAC CC-CC-CC-UU-UU-UU
  - CC = Company identifier
  - UU = Unique identifier
- Insert FF-FE between CC and UU
- CC CC CC FF FE UU UU UU

- MAC = 00-D0-59-0C-61-F3
- EUI = 00-D0-59-FF-FE-0C-61-F3

- FE80::323:45FF:FE67:890A



Multicast Address

11111111 Flag Scope Group ID

FF00::/8 addresses are Multicast Addresses

- Addresses available only for a given scope
  - FF02:0:0:0:0:0:0:1 : All the nodes of the link
  - FF02 :0:0:0:0:0:0: 2 : All the routers of the link
  - FF05 :0:0:0:0:0:0: 2 : All the routers of the site
  - FF02 :0:0:0:0:0:0: D : All the PIM routers of the link
  - …
- Addresses available for all scopes
  - FF0X :0:0:0:0:0:0: 101 : Network Time Protocol (NTP)
  - FF0X :0:0:0:0:0:0: 109 : MTP Multicast Transport Protocol

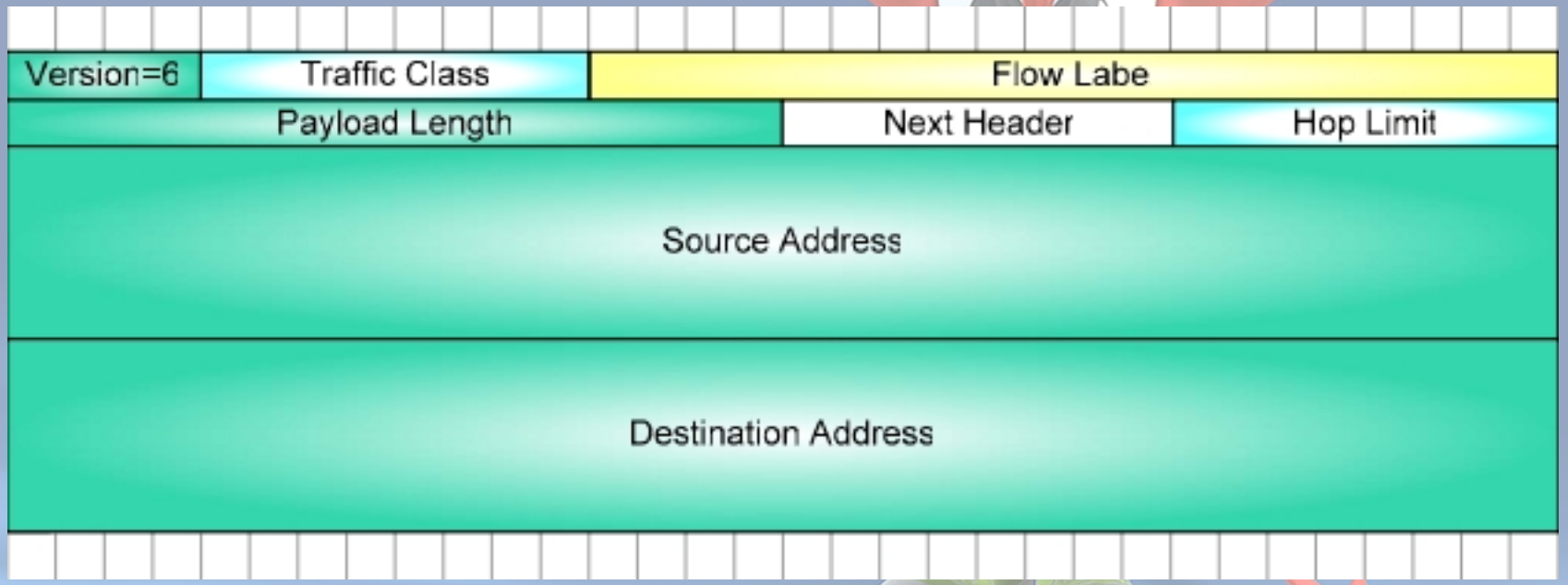
Scope

0: Reserverd
1: Interface Local
2: Link Local
3: Subnet local
4:Admin local
5:Site local
8: Organization Local
E: Global
F: Reserved

IPv4 Header Format

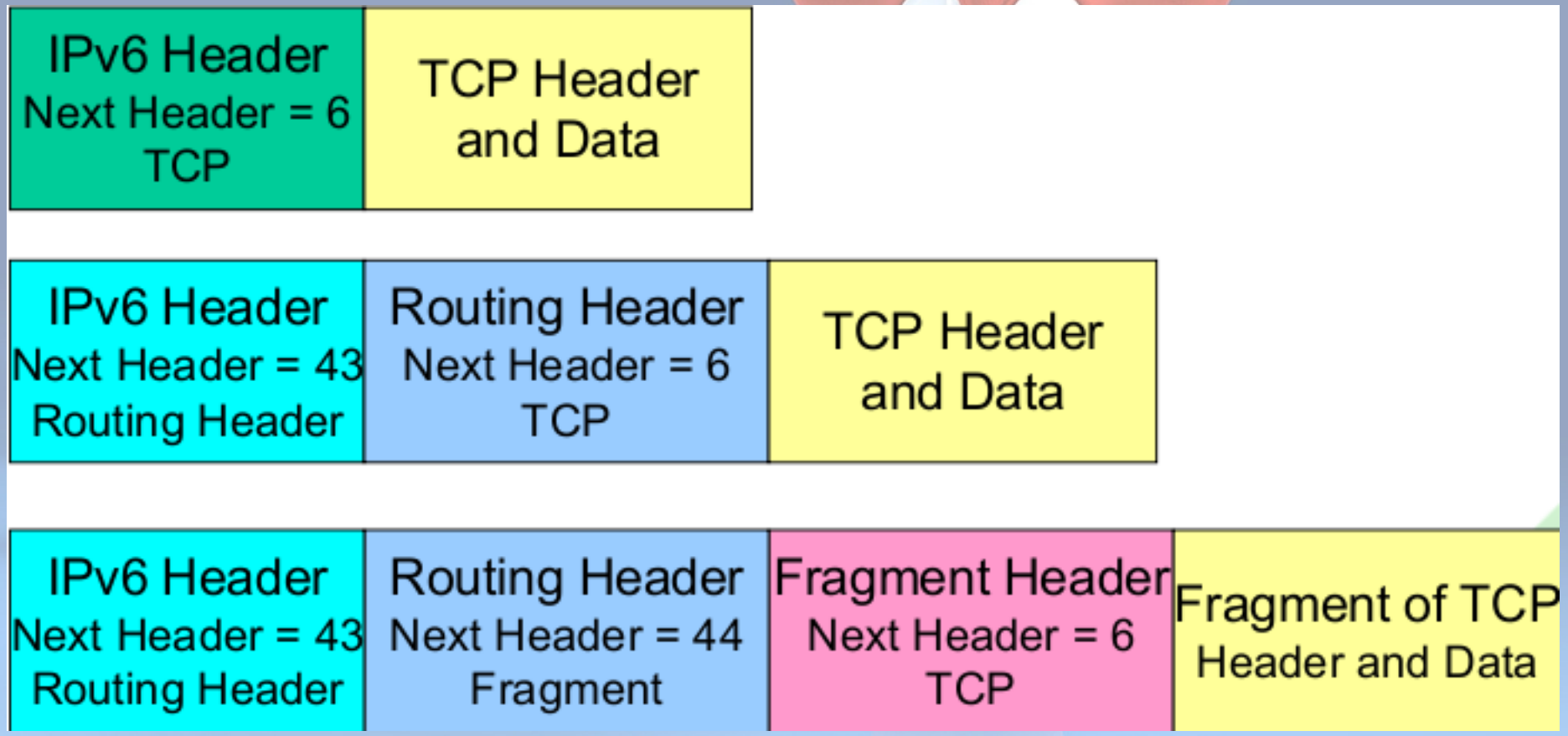
13 Fields

IPv6 Header Format



8 Fields

Extension Headers



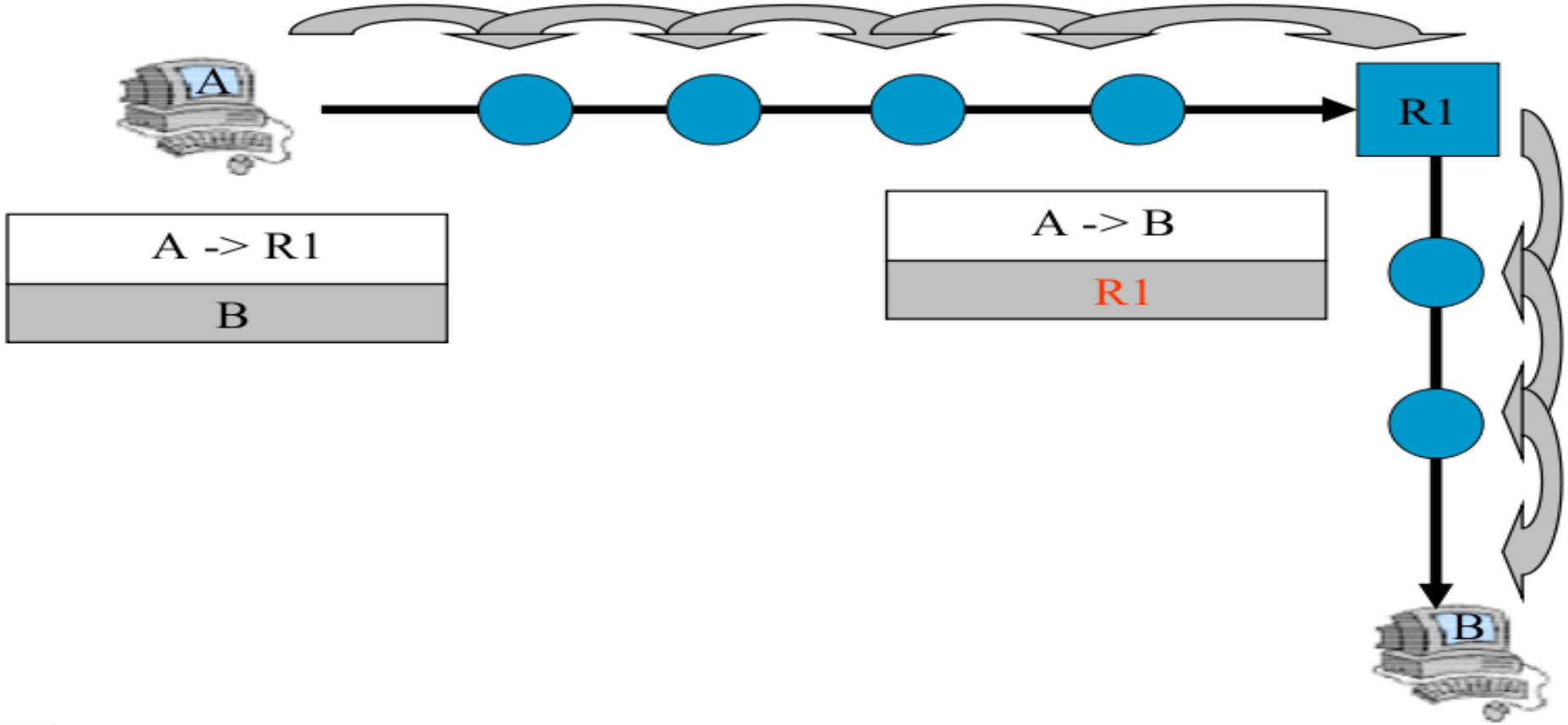
Extension Headers Order

- Hop by Hop (0) => **Processed by every router**
- Destination Option Header (60) =>**Processed by routers listed in Routing extension**

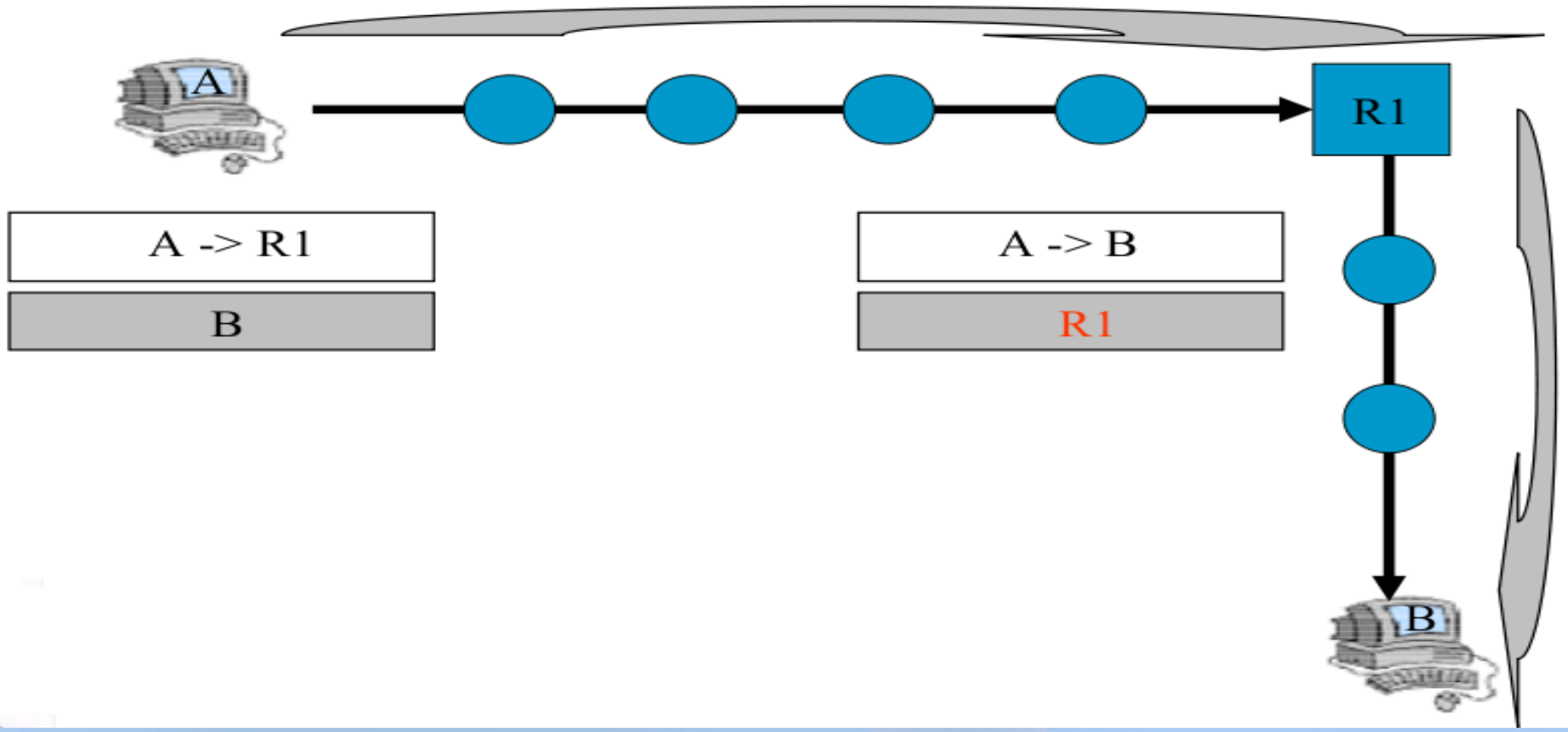
–If Routing Header is used, and D.O for all destinations

- Routing Header (43) =>**List of routers to cross**
- Fragment Header (44) =>**Processed by the destination**
- Authentication Header (51) =>**After reassembling the packet**
- Encapsulation Security Payload Header (50)
- Mobility Header (135)
- Destination Option Header (60) =>**Processed only by the dest.**
- Upper-Layer Header (varies) OR No Next Header (59)

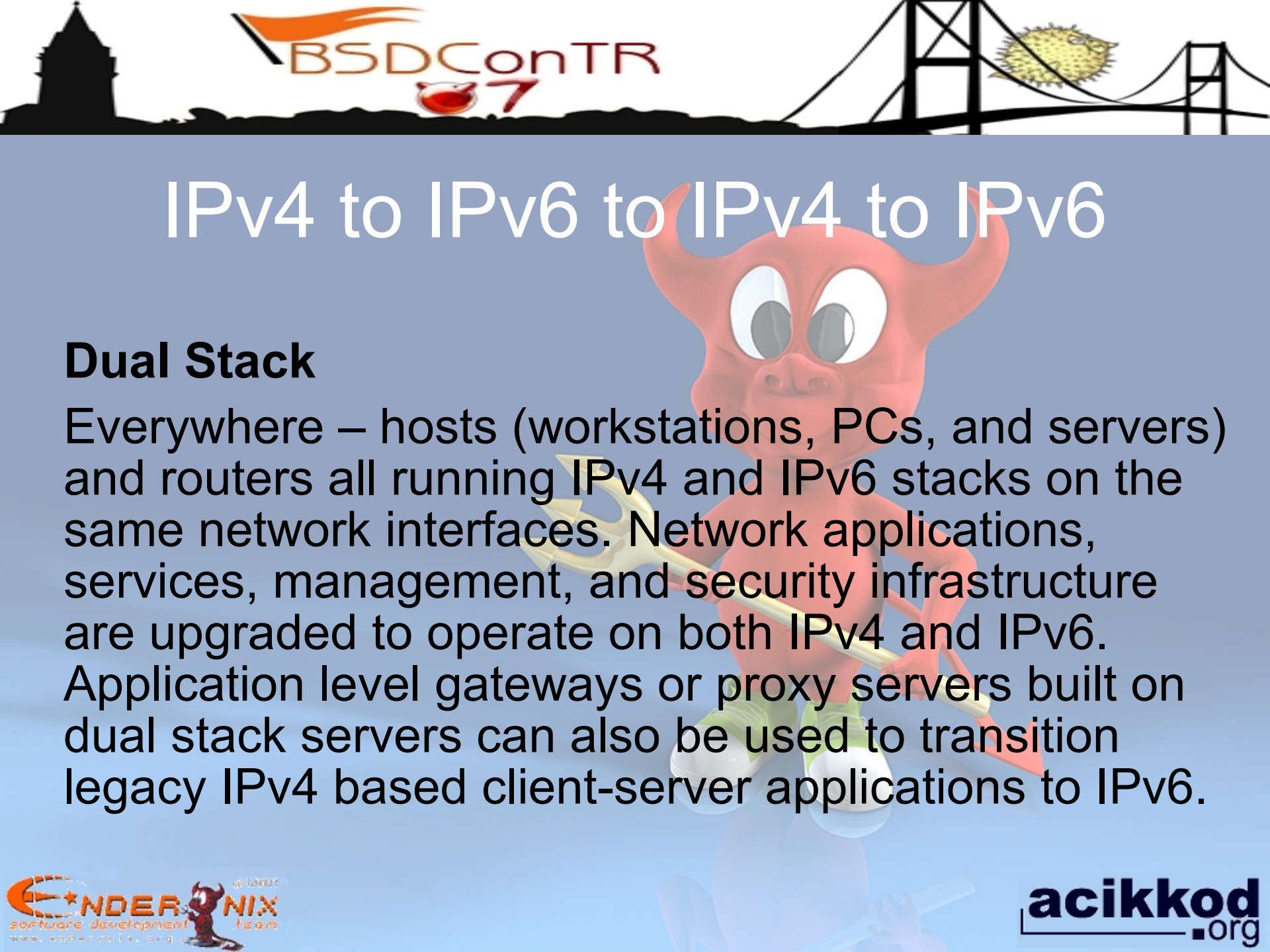
v4 options vs. v6 extensions



v4 options vs. v6 extensions



IPv4 to IPv6 to IPv4 to IPv6



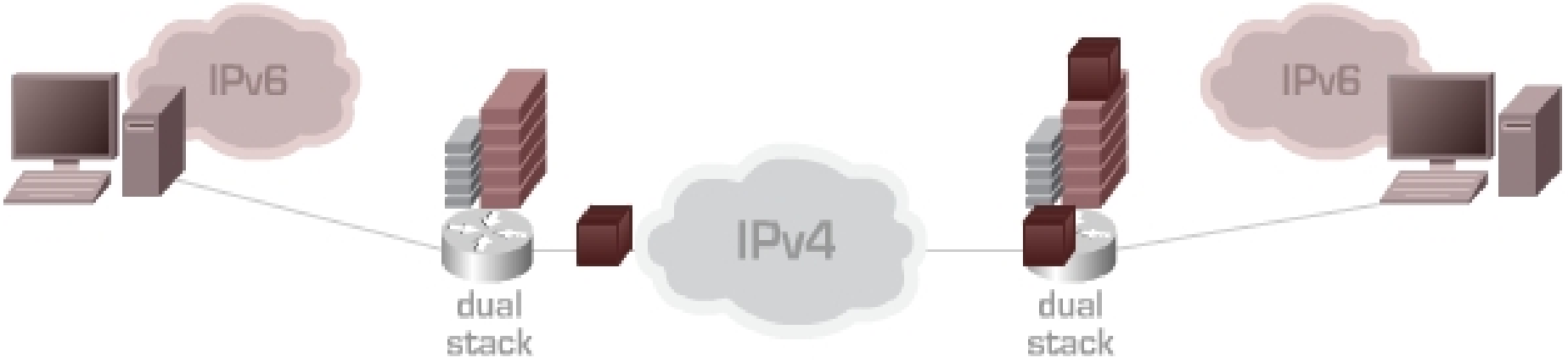
Dual Stack

Everywhere – hosts (workstations, PCs, and servers) and routers all running IPv4 and IPv6 stacks on the same network interfaces. Network applications, services, management, and security infrastructure are upgraded to operate on both IPv4 and IPv6. Application level gateways or proxy servers built on dual stack servers can also be used to transition legacy IPv4 based client-server applications to IPv6.

Dual Stack

IPv4 communication → IPv4 protocol stack

IPv6 communication → IPv6 protocol stack



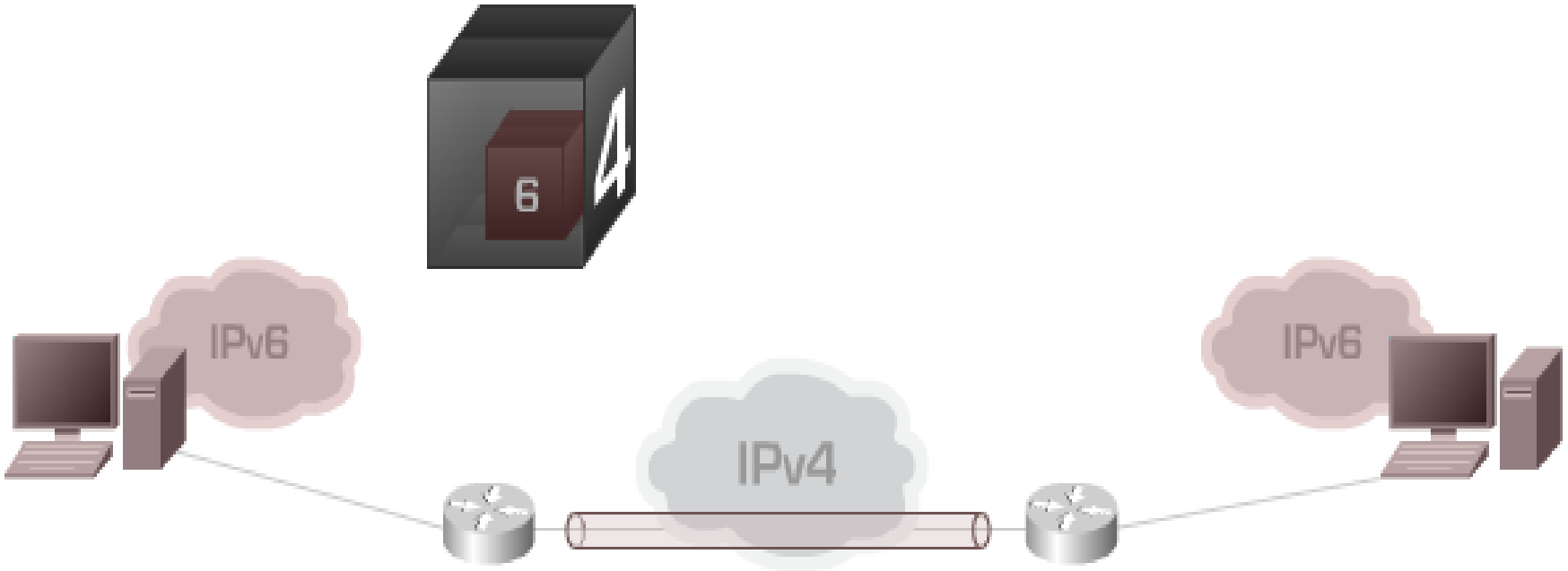
IPv4 to IPv6 to IPv4 to IPv6

Tunneling

Encapsulating IPv6 packets within IPv4 packets for transmission over IPv4-only network infrastructure.
A network infrastructure may also be born “IPv6-only” and tunnel IPv4. Tunneling may be through manually set up tunnels, brokered tunnels2, or numerous automated host-to-router tunneling solutions.
If tunneling is used, an enterprise’s security and network management infrastructure still needs to be upgraded for IPv6.

Tunneling

- **Tunneling** IPv6 packets encapsulated in IPv4 packets



IPv4 to IPv6 to IPv4 to IPv6

Protocol Translation

Translation of IPv4 packets to IPv6 and vice versa, but only as a last resort this is because translation interferes with end-to-end network communications and security. Since most new IPv6 equipment is deployed with a dual stack, the IPv4 side is compatible with legacy IPv4 devices without translation

Transition

applied at three different layers of the protocol stack

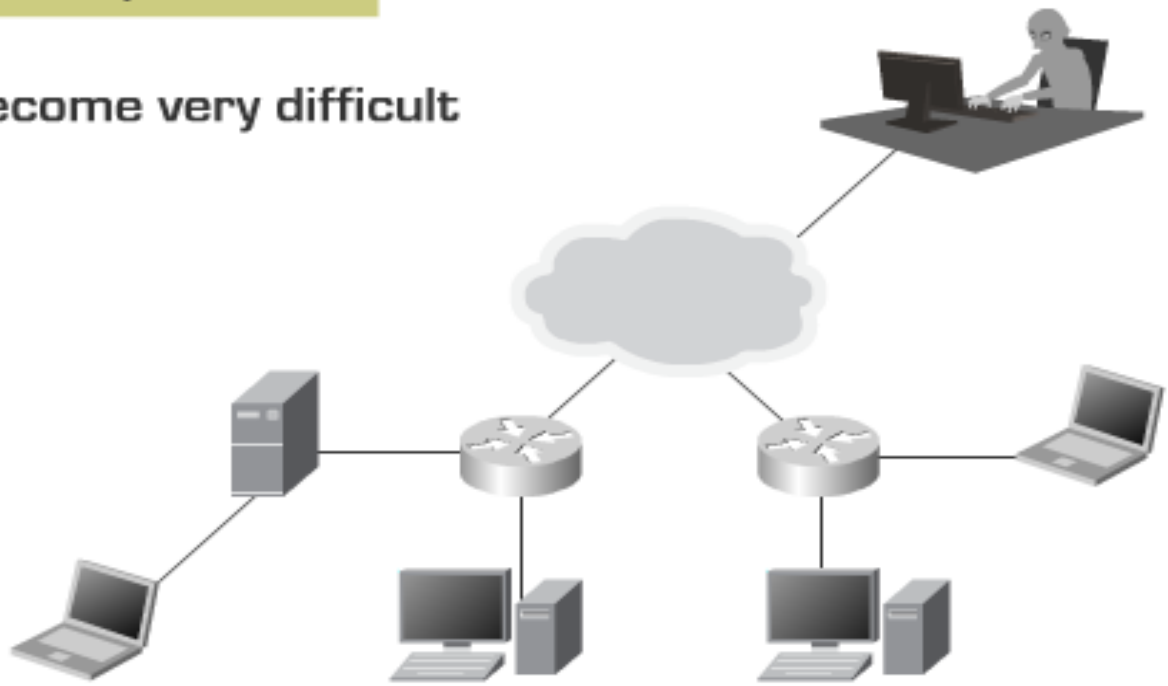
last resort mechanism

Security

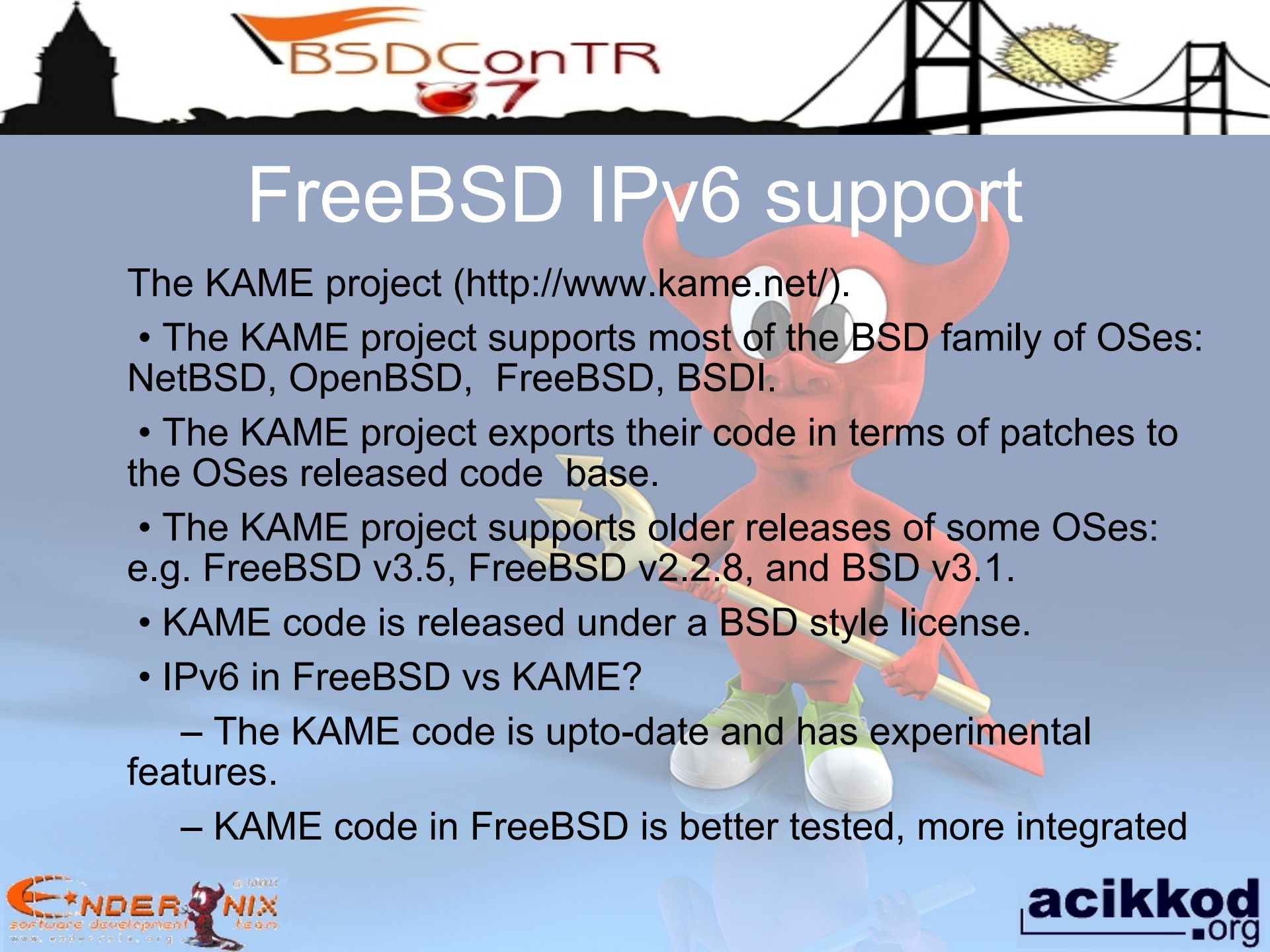
Just One Example :

3FFE:0501:0008:0000:0000:000[illegible]:6CDB:CDCB

exhaustive scan has become very difficult



FreeBSD IPv6 support

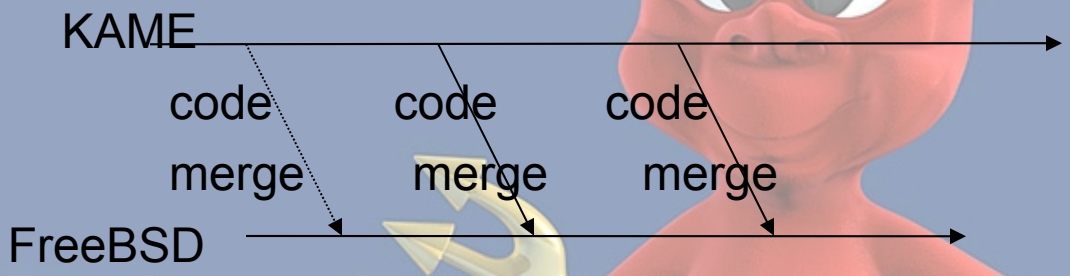


The KAME project (http://www.kame.net/).

- The KAME project supports most of the BSD family of OSes: NetBSD, OpenBSD, FreeBSD, BSDI.
- The KAME project exports their code in terms of patches to the OSes released code base.
- The KAME project supports older releases of some OSes: e.g. FreeBSD v3.5, FreeBSD v2.2.8, and BSD v3.1.
- KAME code is released under a BSD style license.
- IPv6 in FreeBSD vs KAME?
  - The KAME code is upto-date and has experimental features.
  - KAME code in FreeBSD is better tested, more integrated

KAME

FreeBSD supports a dual stack (IPv4 + IPv6).



FreeBSD IPv6 code tracks that of the KAME project.

- IPv4 code is a BSD 4.4 Lite based stack with enhancements and bug fixes.
- the core FreeBSD networking utilities (telnet, ftp) support IPv6.
- the default FreeBSD 6-STABLE install is IPv6 enabled.
- many 3rd party applications (eg:- mpg123, ssh) have been IPv6 enabled by the FreeBSD Ports team and the KAME project working together.

Configuring IPv6

A simple 3-step procedure:

- build and install a kernel with IPv6 capabilities (default GENERIC kernel is already IPv6 capable).
- turn on the knobs in /etc/rc.conf, setup /etc/hosts . . .
- reboot

Building and Installing the kernel

- modify the existing kernel configuration file
  (copy /usr/src/sys/i386/conf/GENERIC and edit to suit)
- add IPv6 related options
- build and install

Kernel Configuration

/usr/src/sys/i386/conf/IPV6ENABLED

```
...
options     INET          #InterNETworking
options     INET6         #IPv6 communications protocols
options     IPSEC         #IP security
options     FFS           #Berkeley Fast Filesystem
...
pseudo-device     gif 4     # IPv6 and IPv4 tunneling
pseudo-device     faith 1 # IPv6-to-IPv4 relaying (translation)
pseudo-device     stf 1     # 6to4 IPv6 over IPv4 encapsulation
...
pseudo-device     bpf       #Berkeley packet filter
```

IPv6 in Ports

The ports mechanism can combine IPv6 patches from the KAME project with other patches needed to otherwise compile the code.

- Ports are classified according to functionality: shells, editors, devel, lang, www, . . .
- Currently there are many applications that use IPv6. These include nearly every popular open source network capable application (e.g. emacs).

– Many network analysis tools (tcpdump, mtr, . . .) are already IPv6 enabled.

Application Ports

FreeBSD supports a sophisticated way to build third party applications (from source) on your local machine.

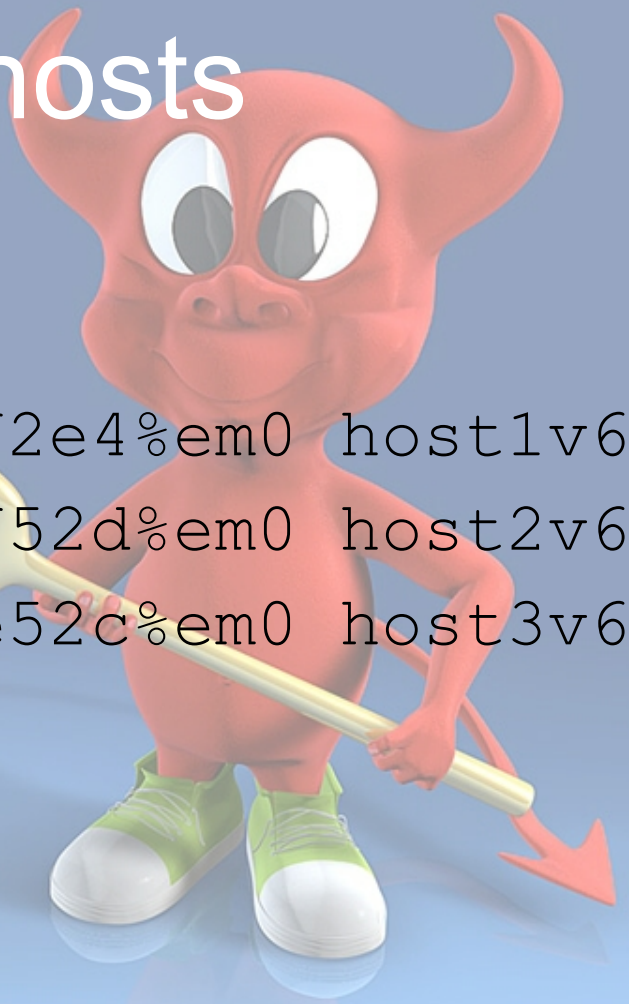
```
ob47191# pwd
/home/ports/net/mtr
ob47191# ls -CF
CVS/ distinfo pkg-comment pkg-plist
Makefile files/ pkg-descr
ob47191# make
>> mtr-0.42.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
>> Attempting to fetch from ftp://ftp.bitwizard.nl/mtr/.
Receiving mtr-0.42.tar.gz (84767 bytes): 100%
84767 bytes transferred in 20.7 seconds (4.01 kBps)
>> mtr-042-v6-20000719.diff.gz doesn't seem to exist
in /usr/ports/distfiles/.
>> Attempting to fetch from ftp://ftp.kame.net/pub/kame/misc/.
Receiving mtr-042-v6-20000719.diff.gz (23526 bytes): 100%
23526 bytes transferred in 5.9 seconds (3.86 kBps)
===> Extracting for mtr-gtk-0.42
```

/etc/rc.conf

```
ipv6_enable="YES" # Set to YES to set up for IPv6.
...
ipv6_defaultrouter="NO"      # Set to IPv6 default gateway
ipv6_gateway_enable="NO"     # host will be a gateway?
ipv6_router_enable="NO"      # run the IPv6 routing
daemon?
rtadvd_enable="YES"          # enable IPv6 router adv.
mroute6d_enable="NO"         # Do IPv6 multicast routing?
gif_interfaces="gif0 gif1" # Examples
stf_interface_ipv4addr=""    # 6to4 IPv6 over IPv4
ipv6_firewall_enable="NO"    # enable IPv6 firewall
...
```

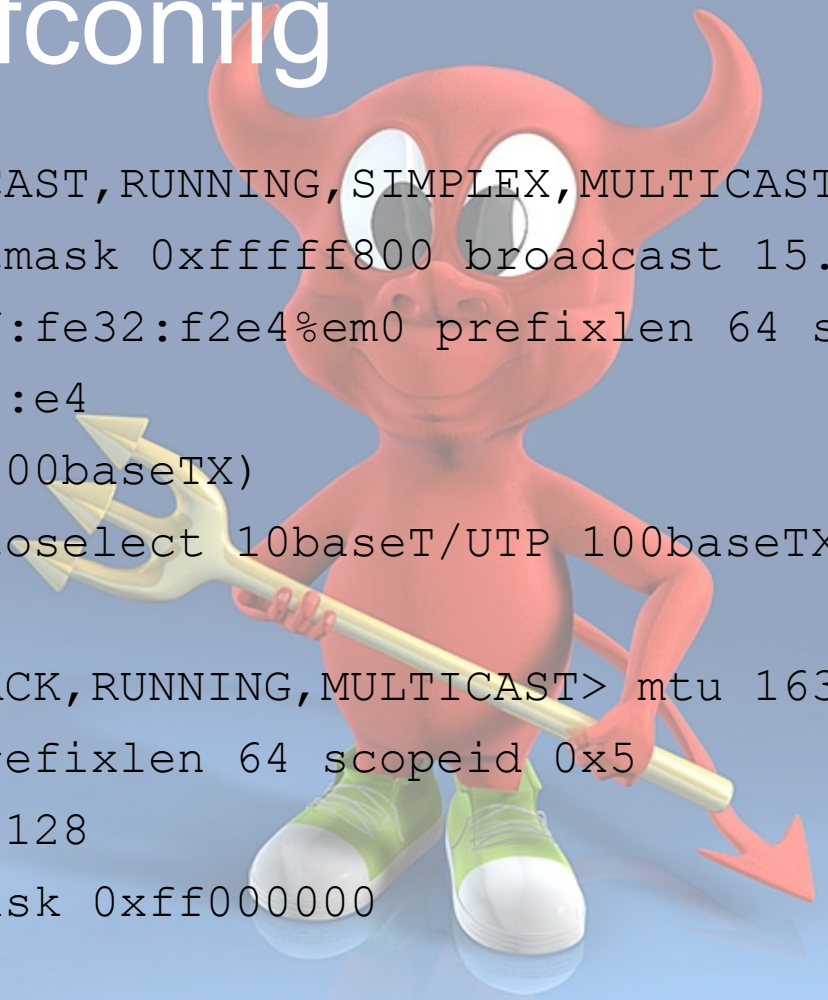
/etc/hosts

```
# IPV6 addresses
fe80::260:b0ff:fe32:f2e4%em0 host1v6
fe80::280:c8ff:fe57:f52d%em0 host2v6
fe80::210:a4ff:fed7:e52c%em0 host3v6
```

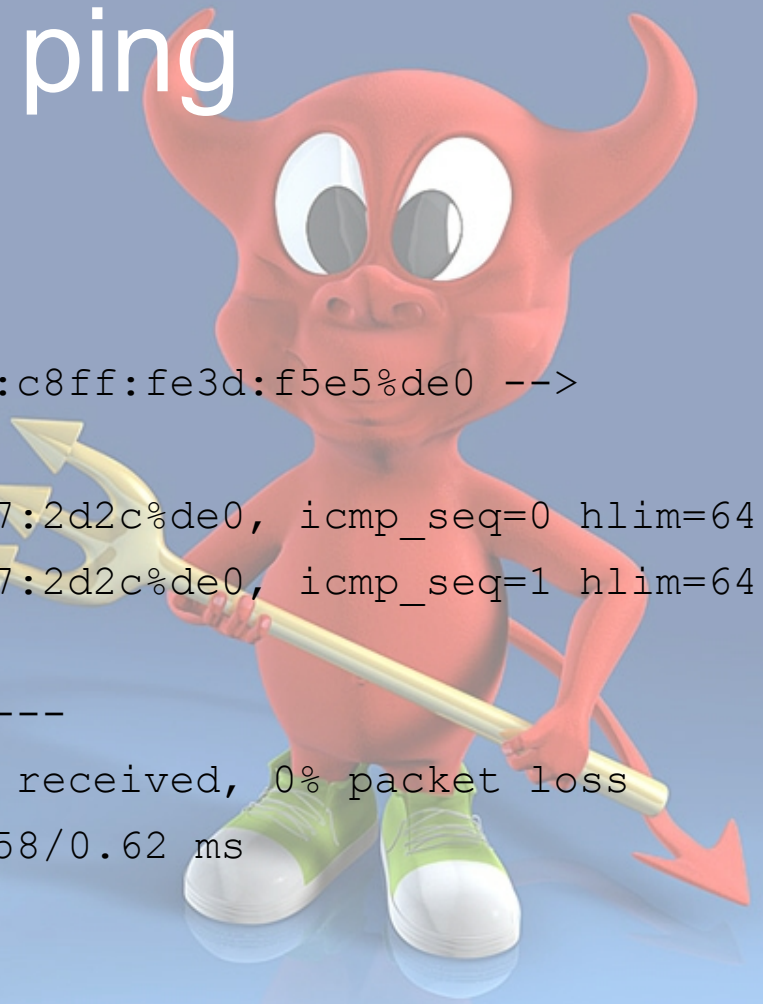


ifconfig

```
$ ifconfig em0
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
     inet 10.10.4.191 netmask 0xfffff800 broadcast 15.10.47.255
     inet6 fe80::260:b0ff:fe32:f2e4%em0 prefixlen 64 scopeid 0x7
     ether 00:10:a4:32:f2:e4
     media: autoselect (100baseTX)
     supported media: autoselect 10baseT/UTP 100baseTX
$ ifconfig lo0
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
     inet6 fe80::1%lo0 prefixlen 64 scopeid 0x5
     inet6 ::1 prefixlen 128
     inet 127.0.0.1 netmask 0xff000000
$ ifconfig gif0
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
```



ping



```
Ping:
ping6 host3v6
PING6(56=40+8+8 bytes) fe80::280:c8ff:fe3d:f5e5%de0 -->
   fe80::260:b0ff:fe32:f2e4%em0
16 bytes from fe80::210:a4ff:fe07:2d2c%de0, icmp_seq=0 hlim=64 time=0.62 ms
16 bytes from fe80::210:a4ff:fe07:2d2c%de0, icmp_seq=1 hlim=64 time=0.54 ms
^C
--- ob47191-v6 ping6 statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.54/0.58/0.62 ms
```

Tunnel Interface

```
ifconfig gif0 create

ifconfig gif0 tunnel 212.156.115.8 64.71.128.82

ifconfig gif0 inet6 2001:470:1F00:FFFF::FD9
2001:470:1F00:FFFF::FD8 prefixlen 128

route -n add -inet6 default 2001:470:1F00:FFFF::FD8

ifconfig gif0 up
```

```
# ifconfig gif0
gif0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST>
mtu 1280
        tunnel inet 212.156.115.8 --> 64.71.128.82
        inet6 2001:470:1f00:ffff::fd9 -->
2001:470:1f00:ffff::fd8
prefixlen 128
        inet6 fe80::212:79ff:fe65:45c8%gif0 prefixlen
64 scopeid 0x8
```

6to4 tunneling

FreeBSD supports automatic 6to4-tunnelling with the **stf** tunnel-interface

Create the interface:

```
ifconfig stf0 create
```

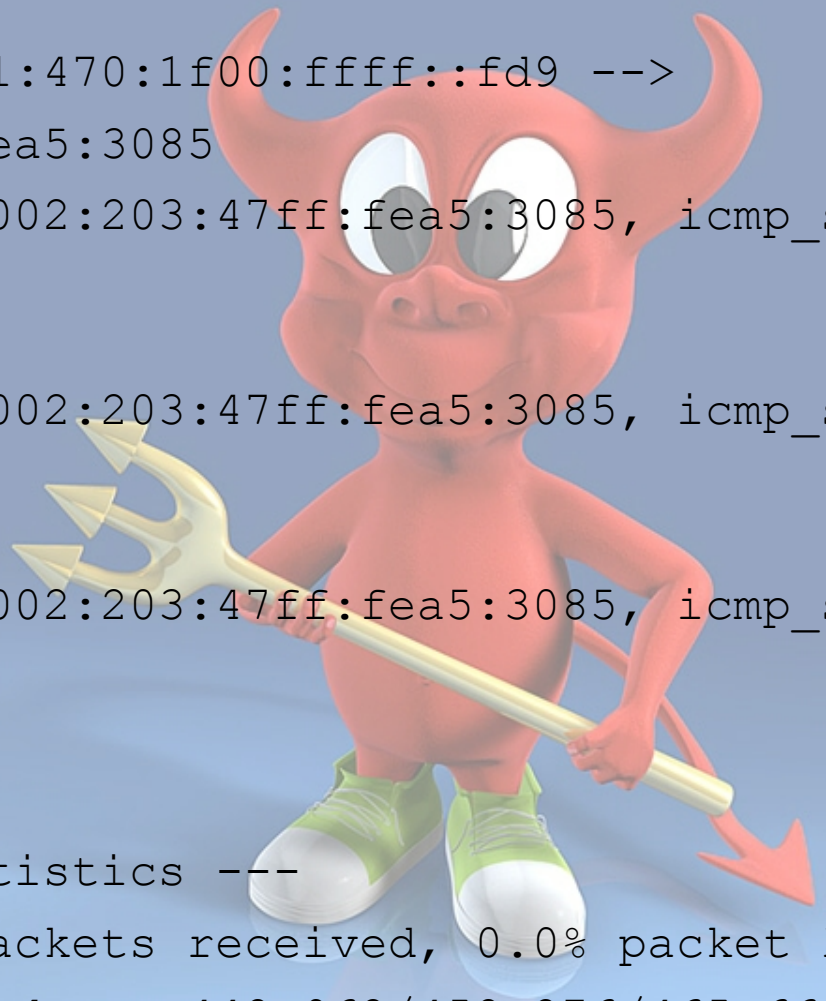
Set up the 6to4/IPv6-endpoint of the tunnel:

```
ifconfig stf0 inet6 $6to4 prefixlen 16 alias
```

Set the (IPv6) default route on the public 6to4 anycast address:

```
route add -inet6 default 2002:c058:6301::
```

```
ping6 www.kame.net
PING6(56=40+8+8 bytes) 2001:470:1f00:ffff::fd9 -->
2001:200:0:8002:203:47ff:fea5:3085
16 bytes from 2001:200:0:8002:203:47ff:fea5:3085, icmp_seq=0
hlim=56
time=442.962 ms
16 bytes from 2001:200:0:8002:203:47ff:fea5:3085, icmp_seq=1
hlim=56
time=465.664 ms
16 bytes from 2001:200:0:8002:203:47ff:fea5:3085, icmp_seq=2
hlim=56
time=453.828 ms
^C
--- www.kame.net ping6 statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/std-dev = 442.962/452.276/465.664/8.046 ms
```



Questions?

Thanks