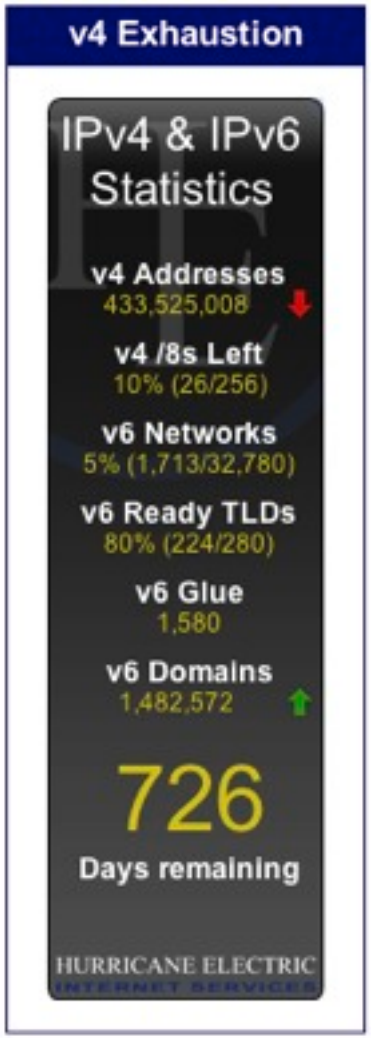
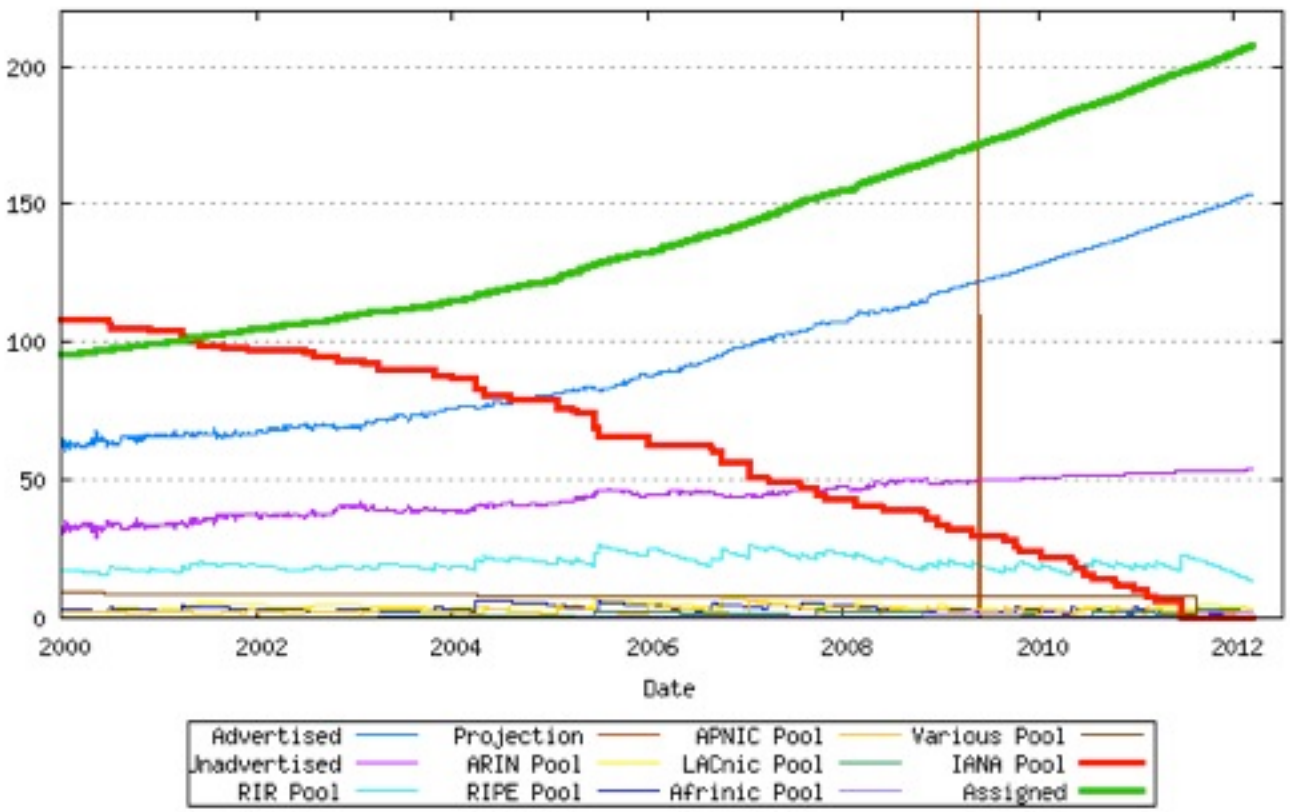
HURRICANE ELECTRIC
INTERNET SERVICES

Porting to Dual Stack -- Not that hard

Owen DeLong
owend@he.net

Why is this important?





Apologies in advance for the Text-Fest

Text Text Text Text Text
Text
Text
Text Text Text Text
Text
Text Text Text Text
Text
Text
Text Text Text
Text
Text

Summary of Porting Steps

- Sample code and other resources available at: http://owend.corp.he.net/ipv6/
- Change variable names when changing types.(e.g. `dest_sin` -> `dest6_sin`)
- Look for old variable name(s) as markers for code to be updated.
- Compile->Repair->Recompile (iterative)
- Test->Debug->Retest (iterative)

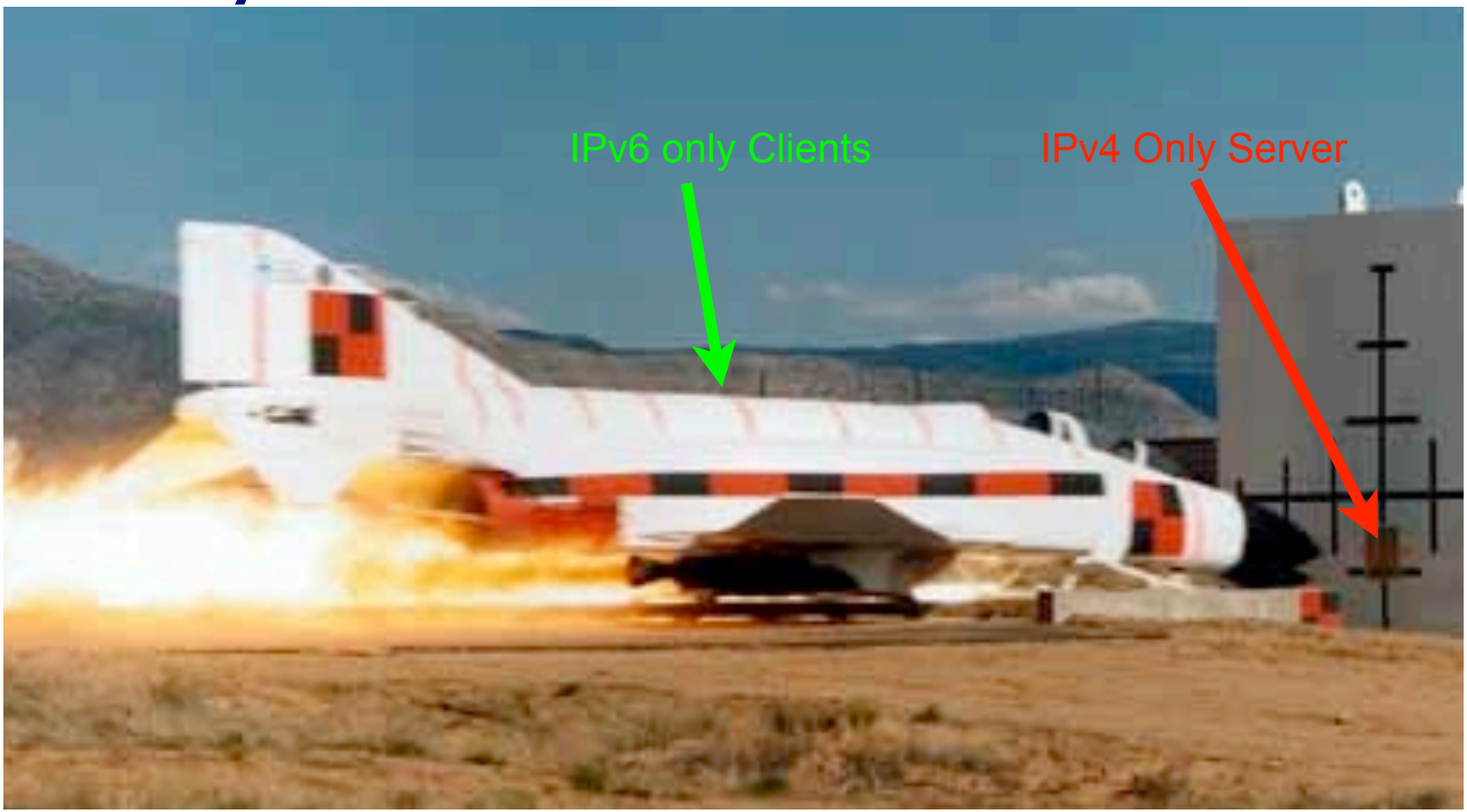
General Changes (IPv4 to dual stack)

- `AF_INET` -> `AF_INET6`
- `sockaddr_in` -> `sockaddr_in6`, `sockaddr_storage` (Generic storage type)
- Same structure members, similar constants, mostly just the address size changes.
- If necessary, check address scoping (link local vs. global and interface scope for link locals)

Some possible gotchas not covered in the examples

- IP Addresses in logs
- IP Addresses stored in databases
- Parsing or other routines that need to deal with IP addresses (use library functions if at all possible)

What happens if we aren't ready?



PERL Porting Example

- Refer to the Source Code Examples
- v4_* are IPv4 only code
- v6_* are same applications ported to dual stack
- Did not rename most variables in this example. (Small codebase, not as important)

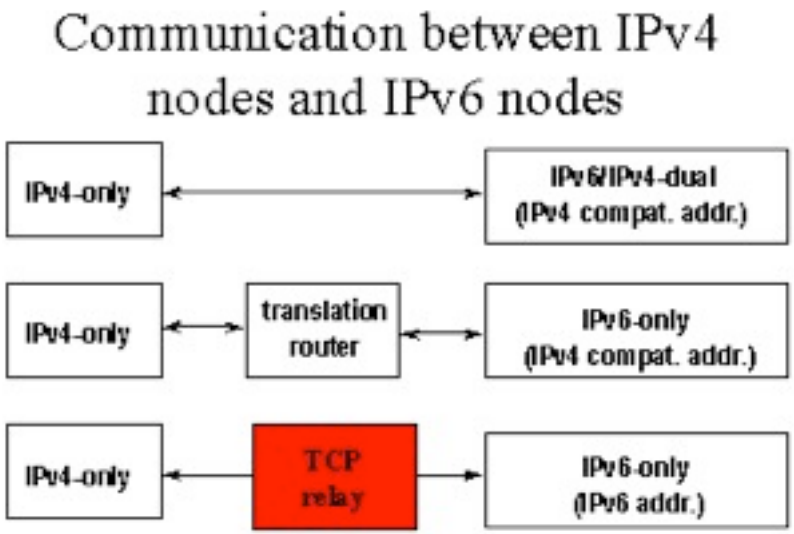
Server Differences (PERL)

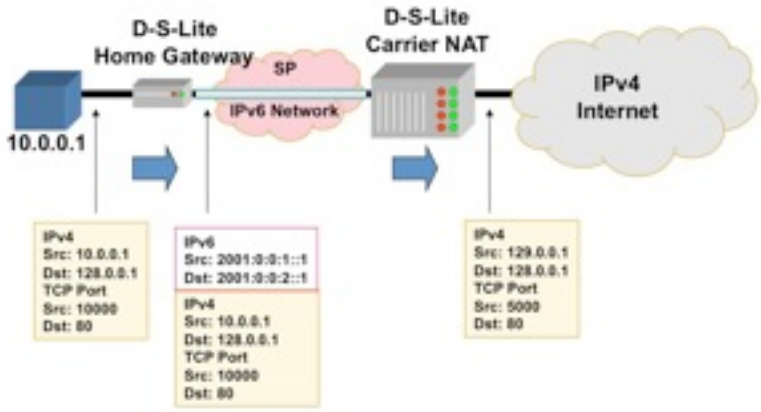
- Add `Socket6` to the modules "`used`" (you still need `Socket`, too). PERL documentation for `Socket6` is minimal and examples limited.
- Gut and replace `get*byname` calls (more on this next slide)
- Change protocol and address families in `socket()` and `bind()` calls.
- Minor changes to processing incoming connections (mostly related to name/address display).

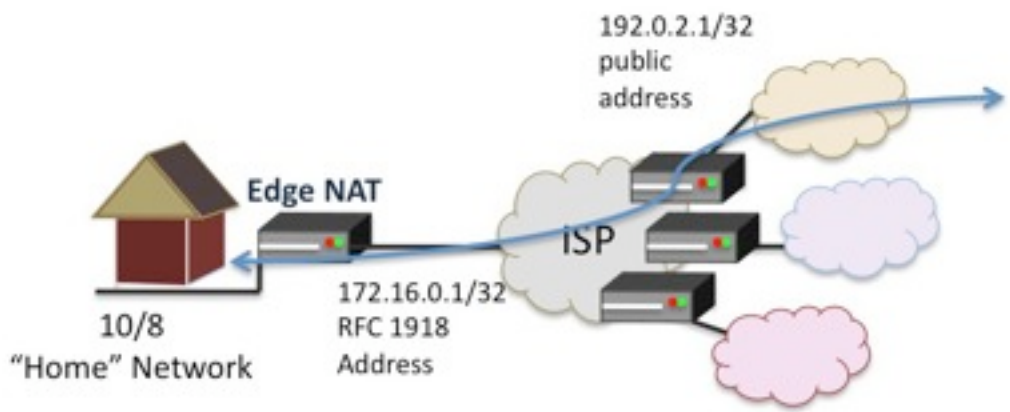
Server Differences (PERL) (cont.)

- Biggest change is conversion from `get*byname()` to `getaddrinfo()`
- Similar changes to C port (same underlying library changes)
- C `getaddrinfo()` returns linked list. PERL `getaddrinfo()` returns straight list (multiple of 5 elements, each 5 elements is a list entry).
- Gotcha on `getaddrinfo()` -- passing in `in6addr_any` does not return `in6addr_any`.

No, really, what happens?







Code Changes (PERL)

- Old way (getservbyname()):

```
my $tcp = getprotobyname('tcp');
my $tcpport = getservbyname('demo', 'tcp');
```

- New way (getaddrinfo()):

```
my ($fam, $stype, $tcp, $saddr, $cname);
my @res = getaddrinfo(in6addr_any(), 'demo', AF_UNSPEC, SOCK_STREAM);
my ($tcpport, $addr);
die "$0: Could not get protocol information" unless @res;
# THis is ugly, but, seems to be necessary to bind to IPv6.
$fam = 0;
($fam, $stype, $tcp, $saddr, $cname, @res) = @res while $fam != AF_INET6;
die "$0: IPv6 unsupported on this system.\n" unless ($fam == AF_INET6);
($tcpport, $addr) = unpack_sockaddr_in6($saddr);
$addr = in6addr_any();
$saddr = pack_sockaddr_in6($tcpport, $addr);
```


Code Changes (PERL) (Cont.)

- IPv4 only:

```
socket(TCPServer, PF_INET, SOCK_STREAM, $tcp) ||
                    die "$0: Could not create socket: $!";
bind(TCPServer, sockaddr_in($tcpport, INADDR_ANY)) ||
                    die "$0: Bind failed: $!";
```

- IPv4/v6 Dual Stack:

```
socket(TCPServer, PF_INET6, SOCK_STREAM, $tcp) ||
                    die "$0: Could not create socket: $!";
bind(TCPServer, $saddr) || die "$0: Bind failed: $!";
```


Code Changes (PERL) (Cont.)

- IPv4 only:

```
my ($port, $iaddr) = sockaddr_in($paddr);
my $name = gethostbyaddr($iaddr, AF_INET);
debug(5, "TCP Connection from $name [".inet_ntoa($iaddr)."] at port $port.\n");
$CLIENTS{$CLIENT} = inet_ntoa($iaddr)."/".$port;
```

- IPv4/v6 Dual Stack:

```
my ($port, $iaddr) = unpack_sockaddr_in6($paddr);
my ($name, $svc)  = getnameinfo($paddr);
debug(5, "TCP Connection from $name [".inet_ntop(AF_INET6, $iaddr).
                        "] at port $port.\n");
$CLIENTS{$CLIENT} = inet_ntop(AF_INET6, $iaddr)."/".$port;
```


PERL Client Migration

- Similar changes to C client
- Add module Socket6 (just like the server)
- Rearrange the address resolution stuff for `getaddrinfo()`
- Add some handling for `AF_INET6` to the connection loop
- Convert `inet_ntoa()` to `inet_ntop()` calls.
- Handle Protocol Family for `socket()` call

Code Changes (PERL) (Cont.)

- IPv4 only:

```
my $tcp = getprotobyname('tcp');
my $tcpport = getservbyname($port, 'tcp');
...
my ($name, $aliases, $addrtype, $length, @addrs) = gethostbyname($server);
die("$0: gethostbyname error: $!\n") if ($?);
die("Invalid server specified.\n") unless(@addrs);
socket(SOCKFD, PF_INET, SOCK_STREAM, $tcp) || die "Couldn't create socket: $!\n";
SOCKFD->autoflush(1);
```

- IPv4/v6 Dual Stack:

```
my @res = getaddrinfo($server, 'demo', AF_UNSPEC, SOCK_STREAM, 'tcp');
die("Could not resolve $server or service demo: ".$res[0].".\n")
                        unless(scalar(@res) >= 5);
```

- Note: In IPv4, socket can be recycled for multiple connects. IPv4/v6 Dual Stack, not so due to possible family change (PF_INET/PF_INET6)

Code Changes (PERL) (Cont.)

- IPv4 only:

```
while (@addrs)
{
  $a = shift(@addrs);
  print "Trying host ", inet_ntoa($a),".\n";
  $dest_sin = sockaddr_in($tcpport, $a);
  last if(connect(SOCKFD, $dest_sin));
  print "Failed to connect to ", inet_ntoa($a), ".\n";
  $dest_sin = -1;
}
```


Code Changes (PERL) (Cont.)

- IPv4/v6 Dual Stack:

```
my ($fam, $stype, $proto, $saddr, $cname);
my ($port, $addr);
while (scalar(@res) >= 5)
($fam, $stype, $proto, $saddr, $cname, @res) = @res;
next unless($saddr);
$cname = $server unless $cname;
print "Unpacking $cname...";
($port, $addr) = ($fam == AF_INET6) ?
                unpack_sockaddr_in6($saddr) : sockaddr_in($saddr);
$addr = inet_ntop($fam, $addr);
print "Trying host $cname ($addr) port $port.\n";
my $PF = ($fam == AF_INET6) ? PF_INET6 : PF_INET;
socket(SOCKFD, $PF, SOCK_STREAM, $proto) || die "Couldn't create socket: $!\n";
SOCKFD->autoflush(1);
last if(connect(SOCKFD, $saddr));
close SOCKFD;
print "Failed to connect to $cname ($addr): $!.\n";
$saddr = -1;
```

- This isn't as bad as it looks. Need better libraries?

Remember this classic? (Apologies to DRC)

Quiz Time!

You're driving towards an invisible brick wall at 60 MPH, what do you do?

A. Accelerate. It will encourage brick wall removal.
B. Set cruise control and close your eyes. "We'll all hit the wall at once".
C. Slow down. Figure out if maybe getting out of the car makes sense.

Function Replacement Guide (all languages)

Old Function	Current Function
get*by*()	getaddrinfo(), getnameinfo()
socket()	socket()†
bind()	bind()†
listen()	listen()
connect()	connect()†
recv*()	recv*()†
send*()	send*()†
accept()	accept()
read()/write()	read()/write()
inet_ntoa()/inet_aton()	inet_ntop()/net_pton() or getnameinfo()†
† parameters change for IPv6 suport	

Structure Replacement Guide

Old Structure	Current Structure
`sockaddr_in,sockaddr_storage†`	`sockaddr_in6,sockaddr_storage†`
`in_addr, int` (Don't do this, even in v4 only)	
`hostent`	`addrinfo`
`servent`	
†sockaddr_storage is a pointer type only can point to either actual type.	

Q&A

Copy of slides available at:
http://owend.corp.he.net/ipv6/PortMeth.pdf

Contact:

Owen DeLong
IPv6 Evangelist
Hurricane Electric
760 Mission Court
Fremont, CA 94539, USA
http://he.net/

owend at he dot net
+1 (408) 890 7992