Make PingER IPV6 compliant

Introduction

There is a simple introduction to IPv6 from Juniper. Basically it increases the address space from 32bits(IPv4) to 128bits (IPv6). We are running out of IPv4 addresses especially in Asia. For sometime now, at SLAC we have only been buying network equipment that is IPv6 compliant. However there is a big step from the network being able to carry IPv6 to actually using it. For example at the network layer one needs IP Address Management (IPAM) tools that support IPv6 so one can track/manage assigning IPv6 addresses and support DHCPv6, DNSv6 etc.

The even bigger problem, however, is the applications that rely on IPv4. Just to take one dear to our heart the PingER archiving analysis relies on IPv4 addresses. Thus as we move to IPv6 we will need to modify PingER to accept both IPv4 and IPv6 addresses.

The US Federal government have said that Federal sites must have outward facing services (web, DNS, email etc) working with IPv6 October 2012, and inward facing services working on IPv6 by October 2014. SLAC is not a Federal site it is a contractor (Stanford) operated site. The DoE has taken a lead in converting its sites to IPv6, but that has not mandated a conversion at SLAC. However in March 2013, SLAC started a project to stand up an IPv6 subnet, enhance the SLAC main web server and email server to be IPv6 compliant, update the DNS records (forward and reverse), put together a CyberSecurity Risks and Mitigations document, get management to accept the risk and open up the web and email services to the Internet. We plan to complete this by October 2013.

Steps

One of the first steps will be to choose a perl module to support authentication of IPv6 addresses. Examples possibly include:

- RegExp::IPv6. There a couple of possibly useful subroutines in traceroute.pl (obtainable here):
  - ```
    sub valid_ip {
      #Given a string, it checks whether it is a valid IP name (returns "n"),
      #or valid IPv4 address (returns "4"), or valid IPv6 address returns "6").
      #If invalid it returnr IP name see: http://stackoverflow.com/questions/106179/regular-expression-
    to-match-hostname-or-ip-address
      #For IPv4 address see: http://answers.oreilly.com/topic/318-how-to-match-ipv4-addresses-with-
    regular-expressions/
      #For IPv6 address see: http://forums.dartware.com/viewtopic.php?t=452
      #Test cases for IPv6 address (we are using aeron) see http://download.dartware.com/thirdparty
    /test-ipv6-regex.pl
      #Examples:
      if(length($_[0])<256) {
        if($_[0]=~/^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9]
    [0-9]?)$/){#IPv4 address
          return "4";
        }
        elsif($_[0]=~/^(((?=(?>.*?::)(?!.*::)))(::)?([0-9A-F]{1,4}::?){0,5}|([0-9A-F]{1,4}:){6})(\2([0-
    9A-F]{1,4}(::?|$)){0,2}|((25[0-5]|(2[0-4]|1[0-9]|[1-9])?[0-9])(\.|$)){4}|[0-9A-F]{1,4}:[0-9A-F]
    {1,4})(?<![^:]:)(?<!\.)\z/i) {#IPv6 Address
          return "6";
        }
        elsif($_[0]=~/^([a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9\-]{0,61}[a-zA-Z0-9])(\.([a-zA-Z0-9]|[a-zA-Z0-
    9][a-zA-Z0-9\-]{0,61}[a-zA-Z0-9]))*$/) {#Name
        #if($_[0]=~/(([a-z0-9]+|([a-z0-9]+[-]+[a-z0-9]+))[.])+/) {#Name
          return "n";
        }
      }
      return "0";
    }
    ```
    ```

```
sub gethostbyname6 {
  #gethostbyname6 yields the IPv6 or IPv6 address for a host name
  # uses dig to get AAAA address for given name
  # returns IPv6 address if successful else returns ""
  my $name=$_[0];
  my $ipaddr="";
  if($ipv eq "4" && gethostbyname($name)) {
    $ipaddr=gethostbyname($name);
    my ($a1, $b1, $c1, $d1)=unpack('C4',$ipaddr);
    $ipaddr=$a1.".".$b1.".".$c1.".".$d1;
  }
  else {
    my $cmd="dig $name -t AAAA";
    if(!($cmd=~ /^([\w\.:\-\s]+)$/)) {#Untaint Check for valid characters
      print "<font color='red'><b>Invalid (tainted) command = $cmd, traceroute aborted!</b><
/font><br>\n";
      exit 1;
    }
    $cmd=$1; #Untaint $cmd.
    my @result=grep(/\s+AAAA\s+/,`$cmd`);
    foreach my $result(@result) {
      if($result =~ /^;/) {next;}
      my @tokens=split(/\s+/,$result);
      if(defined($tokens[4])) {$ipaddr=$tokens[4]; last;}
    }
  }
  return $ipaddr;
}
```

- I believe the measurements code (pinger2.pl) will work with both IPv4 and IPv6. This was part of an IPv6 project at SLAC many years ago that used the SLAC IPv6 allocation about which there is a presentation. Thus for monitoring sites pinger2.pl should be OK.
- A simple place to start would be traceroute.pl, since it does not require access any external modules, and is conceptually very simple. Some work has been done on this already, and it is in use in perfSONAR. It uses ping6 and traceroute6.
- I suspect ping_data.pl will need some mods also.

Once pinger2.pl, traceroute.pl and pinger_data.pl have been made IPv6 capable then the package that monitoring hosts need to install will be IPv6 capable.

It would be good to have a test bench, e.g. an IPv6 host on which we can debug the modified applications.

Converting PingER analysis etc to IPv6 could be an interesting project for someone who wants to be in the vanguard.

Most of the changes for IPv6 would be required in the analysis phases (see PingER data flow at SLAC for information on the data flow from measurement through gathering and analysis). Thus Anjum was thinking of rewriting the input IPv6 data with the IPv4 address so leaving the analysis as is. Another thought was to remove the IP address altogether.

Probably it would be best to port the gathering, archiving, analysis, prsentation code to another site for modification to support IPv6. For some information on portingthe code see PingER#PortingPingERArchivetoNUST.

Hints

An IPv6 host is ipv6.google.com(2001:4860:800f::68)

You can use http://www.subnetonline.com/pages/ipv6-network-tools/online-ipv6-ping.php to see if a host is reachable with IPv6.

The Linux online manual has a very useful chapter on IPv6-ready test/debug programs