Dual-stack IPv4+IPv6 monitoring with Nagios

Teemu Kiviniemi, CSC/Funet
6th June 2012
6th TF-NOC meeting
Dublin, Ireland
Introduction

World IPv6 Launch is today!

When enabling IPv6 support in services, it is crucial to monitor with both IPv4 and IPv6.

- If the IPv6 service is not monitored, IPv6-related problems may not be detected and IPv6-enabled users will see deteriorated performance.

How to monitor dual-stack service with Nagios?
Monitoring a dual-stack HTTP server

Solution: run `check_http` for the IPv4 and IPv6 services separately.

Problems:
- It’s easy to forget to configure the IPv6 service.
- Doubles the number of monitored services.
- Doubles the number of service notifications.
- Doubles the number of services that must have scheduled downtime during service maintenance.
Monitoring a dual-stack HTTP server

Solution: run check_http for the IPv4 and IPv6 services separately, and use check_cluster to combine the results.

Problems:
- It’s easy to forget to configure IPv6 monitoring.
- Three times the number of monitored services.
- Error-prone to configure manually.
Introducing check_v46

A Nagios plugin wrapper that makes monitoring dual-stack services easier.
Compatible with most Nagios plugins that support the standard command line options: -4, -6, --hostname
Developed at Funet.
How the wrapper works

The wrapper behaves like any Nagios check plugin.

To use, replace

```
check_http -H www.funet.fi
```

with

```
check_v46 check_http -H www.funet.fi
```

The wrapper queries DNS and runs the actual plugin once or twice with -4 or -6 parameters if IPv4/IPv6 addresses exist for the name.
How the wrapper works (continued)

The wrapper returns the worst result returned by the actual check plugin.

- If the IPv4 service is in CRITICAL state, but the IPv6 service is in WARNING state, CRITICAL will be returned by the wrapper.

Any returned performance data labels are prepended with "ipv4_" or "ipv6_".

The extended status information will contain the full output of the executed check plugins.
Benefits

Easy to configure: it is enough to update the configuration of the check command.
- All services monitored with the check command become instantly dual-stack aware.

Monitoring follows the actual data returned by DNS.
- Dual-stack monitoring is done only if the host name has both IPv4 and IPv6 addresses.
- Follows the behaviour of real clients.

Only one monitored service in Nagios.
Disadvantages

- Only one service for both IPv4 and IPv6
  - No separate availability statistics for IPv4 and IPv6.
- May affect performance, as IPv4 and IPv6 service checks are run serially.
PLUGIN WRAPPER DEMO
Demo: HTTP monitoring

$ check_http -4 -H demo.funet.fi
HTTP OK: HTTP/1.1 301 Moved Permanently - 523 bytes in 0.006 second
  response time |time=0.005673s;;;0.000000 size=523B;;0
Demo: HTTP monitoring

$ check_http -4 -H demo.funet.fi
HTTP OK: HTTP/1.1 301 Moved Permanently - 523 bytes in 0.006 second
  response time |time=0.005673s;;;0.000000 size=523B;;;0

$ check_http -6 -H demo.funet.fi
HTTP OK: HTTP/1.1 301 Moved Permanently - 523 bytes in 0.004 second
  response time |time=0.003978s;;;0.000000 size=523B;;;0
Demo: HTTP monitoring

$ check_http -4 -H demo.funet.fi
HTTP OK: HTTP/1.1 301 Moved Permanently - 523 bytes in 0.006 second
  response time |time=0.005673s;;;0.000000 size=523B;;;0
$ check_http -6 -H demo.funet.fi
HTTP OK: HTTP/1.1 301 Moved Permanently - 523 bytes in 0.004 second
  response time |time=0.003978s;;;0.000000 size=523B;;;0
$ check_v46 check_http -H demo.funet.fi
  ipv6_time=0.004730s;;;0.000000 ipv6_size=523B;;;0
  ipv4_time=0.002237s;;;0.000000 ipv4_size=523B;;;0
Status details:
  IPv6/demo.funet.fi:
  HTTP OK: HTTP/1.1 301 Moved Permanently - 523 bytes in 0.005 second
    response time
  IPv4/demo.funet.fi:
  HTTP OK: HTTP/1.1 301 Moved Permanently - 523 bytes in 0.002 second
    response time
Demo: SSH monitoring

$ check_ssh -4 -H demo.funet.fi
SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
Demo: SSH monitoring

```
$ check_ssh -4 -H demo.funet.fi
SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
$ check_ssh -6 -H demo.funet.fi
SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
```
Demo: SSH monitoring

$ check_ssh -4 -H demo.funet.fi
SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
$ check_ssh -6 -H demo.funet.fi
SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
$ check_v46 check_ssh -H demo.funet.fi
Status details:
IPv6/demo.funet.fi:
SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
IPv4/demo.funet.fi:
SSH OK - OpenSSH_5.3p1 Debian-3ubuntu7 (protocol 2.0)
Demo: Ping monitoring

$ check_fping -4 -H demo.funet.fi
FPING OK - demo.funet.fi (loss=0%, rta=0.330000 ms)|loss=0%;;0;100
  rta=0.000330s;;;0.000000
Demo: Ping monitoring

$ check_fping -4 -H demo.funet.fi
FPING OK - demo.funet.fi (loss=0%, rta=0.330000 ms)|loss=0%;;0;100
   rta=0.000330s;;;0.000000
$ check_fping -6 -H demo.funet.fi
FPING OK - demo.funet.fi (loss=0%, rta=0.360000 ms)|loss=0%;;0;100
   rta=0.000360s;;;0.000000

We use patched check_fping
Demo: Ping monitoring

$ check_fping -4 -H demo.funet.fi
FPING OK - demo.funet.fi (loss=0%, rta=0.330000 ms)|loss=0%;;;0;100
    rta=0.000330s;;;0.000000
$ check_fping -6 -H demo.funet.fi
FPING OK - demo.funet.fi (loss=0%, rta=0.360000 ms)|loss=0%;;;0;100
    rta=0.000360s;;;0.000000
$ check_v46 check_fping -H demo.funet.fi
OK: IPv6/demo.funet.fi OK, IPv4/demo.funet.fi OK | ipv6_loss=0%;;;0;100
    ipv6_rta=0.000440s;;;0.000000 ipv4_loss=0%;;;0;100
    ipv4_rta=0.000380s;;;0.000000
Status details:
IPv6/demo.funet.fi:
FPING OK - demo.funet.fi (loss=0%, rta=0.440000 ms)
IPv4/demo.funet.fi:
FPING OK - demo.funet.fi (loss=0%, rta=0.380000 ms)

We use patched check_fping
Demo: Broken IPv6

$ check_fping -4 -H demo.funet.fi
FPING OK - demo.funet.fi (loss=0%, rta=0.310000 ms)|loss=0%;;0;100
    rta=0.000310s;;;0.000000
Demo: Broken IPv6

$ check_fping -4 -H demo.funet.fi
FPING OK - demo.funet.fi (loss=0%, rta=0.310000 ms)|loss=0%;;0;100
 rta=0.000310s;;;0.000000

$ check_fping -6 -H demo.funet.fi
FPING CRITICAL - demo.funet.fi (loss=100% )|loss=100%;;0;100
Demo: Broken IPv6

```bash
$ check_fping -4 -H demo.funet.fi
FPING OK - demo.funet.fi (loss=0%, rta=0.310000 ms)|loss=0%;;0;100
   rta=0.000310s;;;0.000000
$ check_fping -6 -H demo.funet.fi
FPING CRITICAL - demo.funet.fi (loss=100% )|loss=100%;;0;100
$ check_v46 check_fping -H demo.funet.fi
   ipv6_loss=100%;;0;100 ipv4_loss=0%;;0;100
   ipv4_rta=0.000400s;;;0.000000
```

Status details:
IPv6/demo.funet.fi:
FPING CRITICAL - demo.funet.fi (loss=100% )
IPv4/demo.funet.fi:
FPING OK - demo.funet.fi (loss=0%, rta=0.400000 ms)
Demo: IPv4 address removed 😊

$ check_fping -4 -H demo.funet.fi
check_fping: Invalid hostname/address - demo.funet.fi
Usage:
    check_fping <host_address> -w limit -c limit [-b size] [-n number] [-T number] [-i number]
Demo: IPv4 address removed 😊

$ check_fping -4 -H demo.funet.fi
check_fping: Invalid hostname/address - demo.funet.fi
Usage:
check_fping <host_address> -w limit -c limit [-b size] [-n number] [-T number] [-i number]

$ check_fping -6 -H demo.funet.fi
FPING OK - demo.funet.fi (loss=0%, rta=0.430000 ms)|loss=0%;;0;100
  rta=0.000430s;;0.000000
Demo: IPv4 address removed 😊

$ check_fping -4 -H demo.funet.fi
check_fping: Invalid hostname/address - demo.funet.fi
Usage:
    check_fping <host_address> -w limit -c limit [-b size] [-n number] [-T number] [-i number]
$ check_fping -6 -H demo.funet.fi
FPING OK - demo.funet.fi (loss=0%, rta=0.430000 ms)|loss=0%;;0;100
       rta=0.000430s;;0.000000
$ check_v46 check_fping -H demo.funet.fi
OK: IPv6/demo.funet.fi OK | ipv6_loss=0%;;0;100
       ipv6_rta=0.001330s;;0.000000
Status details:
IPv6/demo.funet.fi:
FPING OK - demo.funet.fi (loss=0%, rta=1.330000 ms)
Conclusions

The wrapper makes dual-stack monitoring very easy.

The wrapper plugin is available at:

check_fping IPv6 patch is available at:
http://sourceforge.net/tracker/?func=detail&aid=3490731&group_id=29880&atid=397599
Wrapper command line help

$ check_v46 --help
Usage: check_v46 -H <host> [--use-ipv4|--use-ipv6] [other options] <plugin> [other plugin args]
Options:
- H, --hostname <host|address>
  Hostname or IPv6/IPv4 address to connect to.
- 4, --use-ipv4
  Check IPv4 only.
- 6, --use-ipv6
  Check IPv6 only.
- a, --ipv4-address=a.b.c.d,e.f.g.h
  Check IPv4 using addresses a.b.c.d and e.f.g.h;
  plugin will be run with the address as --
  hostname
  option argument (replacing the original argument of
  --hostname).
- A, --ipv6-address=A::B,C::D
  Check IPv6 using addresses A::B and C::D; see -a
  above
  for notes on plugin --hostname option handling.
<plugin>
  Path to the actual plugin program to run.
<other plugin args>
  Any command line arguments besides -H, -4, -6,
  -a and -A
  will be passed as-is to the <plugin>.

The order of options is not relevant, e.g.
'check_v46 -H <host> <plugin>' is effectively the
same as
'check_v46 <plugin> -H <host>'

Examples:
check_v46 check_ssh -H host.example.org
  "Automatic dual stack test": Run check_ssh for
  IPv6 only if
  the system resolver returns an IPv6 address and
  likewise for
  IPv4.
check_v46 check_ssh -H host.example.org -4
  - Run IPv4 check only.
check_v46 check_ssh -H host.example.org -6
  - Run IPv6 check only.
check_v46 check_ssh -H host.example.org -a a.b.c.d
  - Pass "--hostname a.b.c.d" to check_ssh when
  checking IPv4
  and have IPv6 checks run automatically as in the
  first example.
check_v46 check_ssh -H host.example.org -6 -A A::B
  - IPv6 checking only, skip resolving
  host.example.org and
  just pass "--hostname A::B" to check_ssh.