Enabling IPv6 at FZU - WLCG Tier2 in Prague

Tomáš Kouba, Jiří Chudoba, Marek Eliáš
Fyzikální ústav AV ČR, Na Slovance 2, Prague 182 21, Czech Republic
E-mail: koubat@fzu.cz

Abstract. The usage of the new IPv6 protocol in production is becoming reality in the HEP community and the Computing Centre of the Institute of Physics in Prague participates in many IPv6 related activities. Our contribution presents experience with monitoring in HEPiX distributed IPv6 testbed which includes 11 remote sites. We use Nagios to check availability of services and Smokeping for monitoring the network latency. Since it is not always trivial to setup DNS in a dual stack environment properly, we developed a Nagios plugin for checking whether a domain name is resolvable when using only IP protocol version 6 and only version 4. We will also present local area network monitoring and tuning related to IPv6 performance. One of the most important software for a grid site is a batch system for a job execution. We will present our experience with configuring and running Torque batch system in a dual stack environment. We also discuss the steps needed to run VO specific jobs in our IPv6 testbed.

1. Introduction
The workflow with IPv4 addresses and the main transformation steps of this workflow (installation, monitoring) from IPv4 to IPv6 world was presented during last CHEP in “IPv6 testing and deployment at Prague Tier 2”[1]. We described limitations of the IPv4 setup and sketched a solution that would be possible using IPv6 addressing. Basic steps for setting up network components were also described and we presented FZU IPv6 testbed that was a part of the HEPiX testbed[2].

In this paper we present the evolution of the testbed. We discuss main problems with SLAAC[3] and DHCPv6[6] implementations, monitoring setup for the testbed and services we provide to the HEPiX testbed and we also discuss further steps in building the testbed as a full featured grid site built with IPv6 support.

2. Address configuration
The IPv6 protocol tries to simplify the client network configuration. There are options to automatically configure the clients just from the pieces of information provided by central network elements (central router in our case). The client then generates its IP address, checks the local network for conflicts and, if there is no conflict, it applies this configuration. This kind of setup is not desired in our environment, because static association of host and IP is preferred. Static in this sense means we want to control the way address is assigned and we do not want the IP address to be changed for example when the operating system of the server is reinstalled or network interface card (NIC) is replaced.

We want to avoid any changes of the IP address of services in production mainly for the following reasons:
• TTL of DNS records: expiration of cached DNS queries is just too long;
• small embedded devices: sometimes do not support DNS, but use NTP or syslog services;
• firewall ACLs: cannot be based on DNS names, IP address must be specified;
• external firewall: some of our nodes have special exceptions to access services of some other site; we do not want to bother them with every replaced NIC.

We described our DHCPv6 and Routing advertisement (RA) setup in our previous paper [2] in more detail. Disabling SLAAC depended on the client setup and a misconfigured client was able to generate a SLAAC address on its own. To prevent the clients of using SLAAC, we used the following setup of the router:

```
ipv6 nd prefix 2001:718:1E01:1725::/64 no-advertise
ipv6 nd ra interval 60
```

These commands will make sure that the router broadcasts itself as the default gateway (via link-local address [4] of the router), so that the clients know where to send all packets with destination IP outside of our network. But with this setting (no-advertise) the clients do not receive any information about the local network prefix and thereby they are not able to perform the autoconfiguration to setup an IPv6 address.

The IP address is obtained from DHCPv6. Unfortunately this solution is not perfect either, partially because of the DUID (unique identifier of server) nature:

• DUID is stored in a lease file (somewhere in /var/lib);
• if a lease file exists, dhcpclient always uses DUID stored there;
• in SL6 there are separate instances of dhcpclient per each interface;
• when a NIC is replaced, old DUID-LL is still used:
  – until reinstallation of the machine;
  – this means machines with changed NICs would probably get inaccessible after reinstallation;
  – solution at FZU: delete the lease file when replacing NIC (and update DHCPv6 configuration).

The solution used at FZU gives enough time (between NIC replacement and OS reinstallation) to update the configuration so the server does not get into the situation where it would have different DUID than we have recorded in the DHCPv6 configuration.

3. Jumboframes
We already use jumboframes[5] in production IPv4 network for communication between storage nodes and worker nodes. But in the public IPv4 network we use standard MTU of 1500 bytes, since the possible transition bring several risks. The biggest one is that some nodes on the path to some collaborating institution may not support jumboframes and may have broken support for Path MTU discovery. Such a router should respond to a PMTU discovery packet with ICMP message “fragmentation needed and DF set” by RFC 1191[7] in IPv4, resp. “Packet too big” by RFC 4443[8] in IPv6.

The transition to IPv6 is a convenient opportunity for moving to jumboframes as we need not to touch the production network and try sending jumboframes through the same paths. Our IPv6 testbed uses MTU 9000 and on physical links we use MTU 9192. Since several sites of the HEPiX IPv6 testbed use jumboframes on IPv6, some of the GridFTP data transfers throughout the HEPiX testbed are running using jumboframes.

We also established a monitoring of PMTU functionality in the WAN network between nodes of HEPiX IPv6 testbed as described in the Monitoring section. The tests have shown that there is a PMTU discovery problem on the path from FZU to 4 of 12 tested sites.
4. Monitoring

4.1. Nagios core
The main tool used at FZU for monitoring is Nagios[10]. For IPv6 testbed we run three instances of Nagios (with Livestatus+check_mk extension) on three machines:

- IPv6-only (Livestatus extension needs a workaround with xinetd to work over IPv6 only connections);
- IPv4-only;
- dual-stack (aggregating instance).

All the instances monitor not only FZU IPv6 testbed but also the HEPiX IPv6 testbed. The basic connectivity tests (PING, SSH) work fine over IPv6. SNMPv6 checks using snmpwalk and snmpget need a special format of the address snmpget ipv6://[fec0::aabb:ccdd] and so it is necessary to define custom check command in Nagios.

4.2. GridFTP and MyProxy
One of the regular tests performed with Nagios on grid sites is transferring files with gridftp protocol. This protocol requires authentication with grid proxy and since we want to use the proxy without manual renewal it is necessary to use MyProxy service[11]. The service had not worked properly with IPv6 so we issued a feature request to the developers in collaboration with Keith Chadwick from FNAL.

The support for IPv6 protocol was recently implemented in MyProxy in version 5.8. At FZU this version has been installed from EMI3 middleware distribution on a dual-stack machine myproxy.ipv6.farm.particle.cz and proxy registration and renewal from IPv6-only client has been tested successfully.

4.3. DNS tests
It is quite easy to make a mistake when trying to add IPv6 support to the DNS setup. There can be missing AAAA glue records[9], parent zone with no authoritative server with IPv6 address and so on. We made a plugin for Nagios which tries to resolve a hostname using IPv4-only and IPv6-only resolver to ensure, that the domain names are well resolvable through both environments.

Currently there are still 6 sites whose domains are not resolvable using IPv6 only.

4.4. Tests for Path MTU Discovery
Since there are several risks in deploying jumboframes, we decided to test the PMTU support at least for the paths between FZU and sites of the HEPiX IPv6 testbed. We use two Nagios sensors for PMTU. The first one tries to detect the PMTU value using the tracepath utility and records the value in the result of the sensor. The second one gets the PMTU value of the first sensor through MK Livestatus[15] and tries to send a appropriately long echo-request ICMPv6 packet to the end host. If the echo-response arrives, we know, that the PMTU was detected correctly and thereby PMTU discovery works on that path.

4.5. PerfSONAR
We setup a dualstack PerfSONAR. Tests to the IPv4 and IPv6 PerfSONAR in WLCG are successfully running but, unfortunately, the results are not stable enough to compare the performance of the two IP protocols in our environment.
5. Batch system
One of the critical services of FZU site is the batch system. We use Torque[13] resource manager with Maui[14] scheduler. Our aim was to submit a job to a local IPv6 cluster. The second step was to be able to process jobs submitted via grid to our local IPv6 cluster.

5.1. Local submission
We have successfully tested the submission of a job as a user from a user interface to the Torque system. The job was executed and input and output files were transferred. The main issues encountered were:

- daemon trqauthd used for authentication does not support IPv6;
- Torque server listens only on IPv4 addresses;
- mom daemons require to communicate via IPv4 (we had to configure the system to return IPv4 address of Torque server first);
- file transfers (job input+output) use scp over IPv6.

These issues demonstrate that Torque is not IPv6-ready and we still need to provide IPv4 connectivity (private range of IPv4 addresses is sufficient).

5.2. Grid submission
Figure 2 shows the addition of more components to the batch system as well as more communication channels with the outside world. These channels are needed for a successful execution of a grid job. The most important service is the CREAM CE[16] that accepts job definitions from the grid and submits them as local batch jobs to the Torque server.

The CREAM CE has been configured:

- with public IPv4 address for submission from external world;
- with private IPv4 address for communication with Torque;
- IPv6 address was configured for other protocols;
- with local MySQL database for job records.

We were able to successfully submit a job to our CREAM server with command line tools and obtain the output on our UI via a gridftp transfer.

5.3. CVMFS
In the preparation for future work we have installed CVMFS[17] on our worker nodes. The SQUID proxy used for CVMFS is the same instance we use for packages installation. The CVMFS with standard configuration works and no IPv6-related tweaks were necessary.
6. Future work
With working setup of batch system and with CREAM CE working as gateway service to the batch system we are able to submit grid jobs. In the next step we want to configure our testbed to receive ATLAS jobs and report problems to the WLCG IPv6 task force[18].

7. Conclusion
This paper presented the experience gained during one year of running an IPv6 testbed at FZU. Several problems were encountered when we tried to install and run services (similar to our production services) in IPv6 environment. Although there are workarounds and solutions to these problems and support of IPv6 progressed over last years, it is still not easy to connect IPv6-enabled (or even IPv6-only) services into grid infrastructure (WLCG/EMI in our case).

7.1. Acknowledgements
This work is partially supported by CESNET, project number 482/2013 and Czech Ministry of Education, Youth and Sports (project LG13009).

References
[1] T. Kouba et al., “IPv6 testing and deployment at Prague Tier 2,” J. Phys.: Conf. Series 396 042034.
[2] J. Chudoba, M. Eliáš, L. Fiala, and T. Kouba, ”IPv6 deployment in Tier2 site at FZU in Prague.”, Proceedings of International Symposium on Grids and Clouds, PoS(ISGC 2012)024, 2012.
[3] S. Thomson, T. Narten, T. Jinmei, “IPv6 Stateless Address Autoconfiguration,” http://tools.ietf.org/html/rfc4862.
[4] R. Hinden, S. Deering, “IP Version 6 Addressing Architecture,” http://tools.ietf.org/html/rfc4291.
[5] D. Borman, S. Deering, R. Hinden, “IPv6 Jumbograms,” http://tools.ietf.org/rfc/rfc2675.
[6] The Internet Society, “Dynamic Host Configuration Protocol for IPv6 (DHCPv6),” http://tools.ietf.org/html/rfc3315.
[7] J. Mogul, S. Deering, “Path MTU Discovery,” http://tools.ietf.org/html/rfc1191.
[8] A. Conta, S. Deering, M. Gupta, “Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification,” http://tools.ietf.org/html/rfc4443.

[9] S. Thomson, C. Huitema, “DNS Extensions to support IP version 6,” http://tools.ietf.org/html/rfc1886.

[10] Nagios Enterprises, “Nagios,” http://nagios.org/.

[11] Globus Alliance, “MyProxy,” http://www.globus.org/grid_software/security/myproxy.php.

[12] M. Reale, “HEP!X IPv6 distributed testbed,” https://w3.hepix.org/ipv6-bis/doku.php?id=ipv6:testbed.

[13] Adaptive Computing, “TORQUE Resource Manager,” http://www.adapтивecomputing.com/products/open-source/torque/.

[14] Adaptive Computing, “Maui Cluster Scheduler,” http://www.adapтивecomputing.com/products/open-source/maui/.

[15] Mathias Kettner, “MK Livestatus,” http://mathias-kettner.de/checkmk_livestatus.html.

[16] European Middleware Initiative, “CREAM Computing Element,” http://grid.pd.infn.it/cream/.

[17] P. Buncic, C. Aguado Sanchez, J. Blomer, L. Franco, A. Harutyunian, P. Mato, Y. Yao, “CernVM—a virtual software appliance for LHC applications.”, Journal of Physics: Conference Series, 219/4, 2010.

[18] S. Campana, A. Sciaba, “WLCG IPv6 Task Force,” https://twiki.cern.ch/twiki/bin/view/LCG/WlcgIpv6.