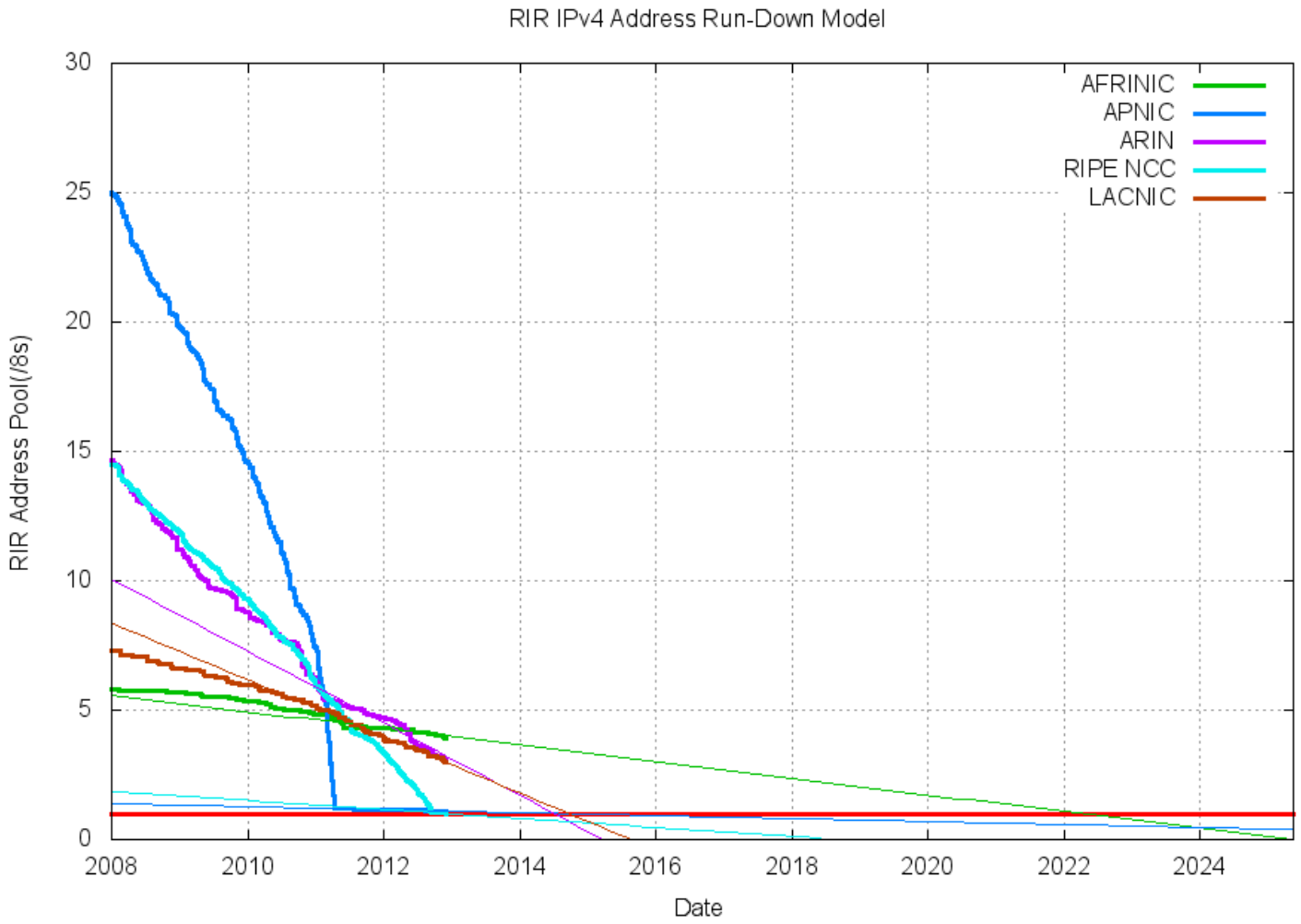
How to sleep tight and keep your applications running on IPv6 transition

The importance of IPv6 Application Testing

About this presentation

- It presents a generic methodology to test the IPv6 functionality of applications
- The methodology is based on Software Testing methodologies and practical experience deploying IPv6
- We present an example of how to test an application

Did you know?



Source: http://www.potaroo.net/tools/ipv4/

IPv6

- Taking off slowly but steady
- Major content providers, large transit providers and network equipment vendors support it
- Slow adoption in access and mobile providers
- OS in users and servers and major applications are ready

But,
Is your application ready for IPv6?

Migrating Applications to IPv6

- Applications made with high level SDKs should work just fine (i.e. Android, iOS, etc.)
- Be aware of hard coded literals (they are bad, very bad practice, even for IPv4)
- Check your datastore needs. Do you need to store IPv4? Then you may need IPv6
  - Remember, 32 vs. 128 bits!

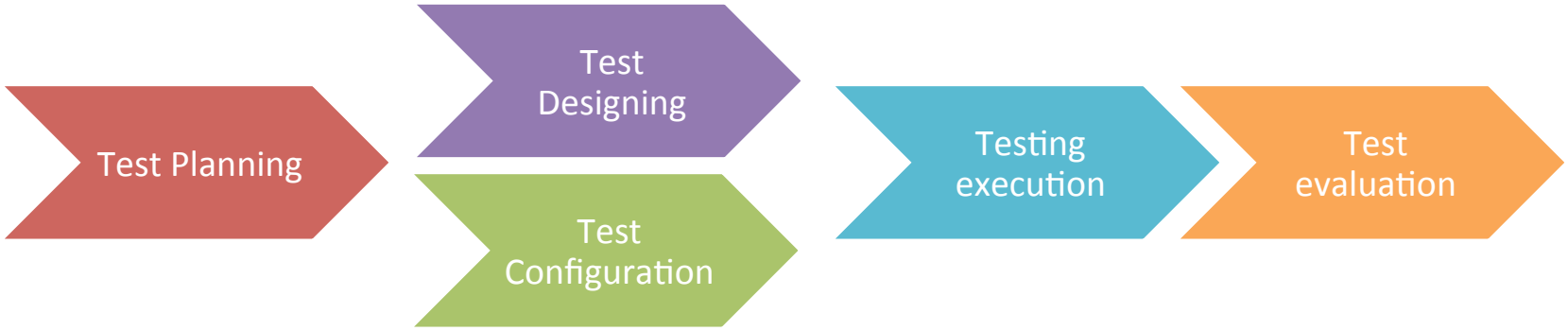
Migrating Applications to IPv6

- If you are programming at “low-level” use dual-stack methods, examples:
  - `socket.inet_ntop` (DS) instead of `socket.inet_ntoa` (IPv4 only) in Python
  - Generic socket instead of `sockaddr_in` in C
  - Java, sockets are automatically ported, but not use `Inet4Address`

Application Testing for IPv6

- If you want to avoid failing to your users when using IPv6 you need to test your apps
- We made a methodology based on software testing principles and our knowledge on IPv6
- The idea was to test extensible and methodologically all our applications before going to production using IPv6

Phases


Test Planning
Test Designing
Test Configuration
Testing execution
Test evaluation

Test Planning

- Decide what are you, and what are you NOT to test
- Be familiar and document the architecture of your system or application
- Identify interfaces and frontiers
- Prioritize functionalities

Test Designing

- You define your testing strategy for each item that you want to test
  - i.e. Planned testing vs. exploratory testing
- Define test cases and experiments
- Some tests:
  - IPv6 parsers, test different IPv6 combinations: full IPv6 addresses, compressed format, try wrong combinations, try ULAs and Global

Test Designing (cont.)

- Some tests (cont.)
  - Communication among components, e.g. database and application server
  - System and environment. Validations, network components
- Validate

Test Configuration

- Test environments; IPv4 only, IPv6 only
  - Isolate and/or filter interfaces to avoid non desired traffic
  - Analyse traffic
  - Disable IPv4 or IPv6 accordingly or plan to use firewalls/filters

Test Configuration (cont.)

- Document test and results
  - Observations, differences between v4 and v6
  - Throughput, delays, problems

Testing Execution

- Try your designed test in IPv4
  - This is your benchmark
  - Detect early problems here, so they won't be "linked" with IPv6
  - Look for long delays and document them
- Try your designed test in IPv6
  - Try exhaustively all your items
  - Verify that IPv4 is not affecting your test

Testing Execution (cont.)

- Try your designed test in IPv6 (cont).
  - Detect long delays and document
  - Document all your response times for further analysis
- If you find problems, fix them and try again (regression tests)

Test Evaluation

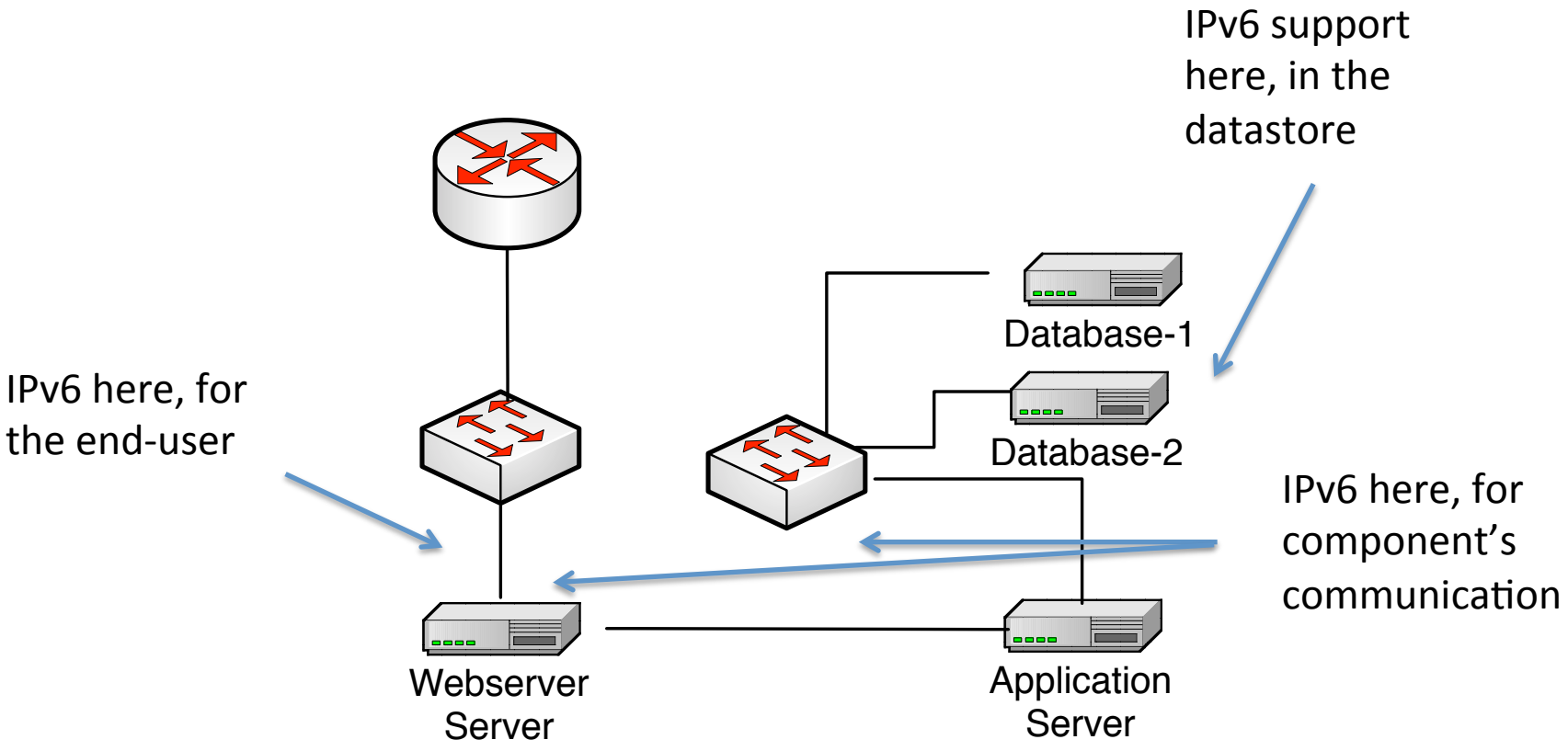
- Collect all the data that your tests generated
- IPv6 Latency, throughput, etc. should be similar than IPv4
- Use your experience for future tests and migration on other applications

Testing an application to manage IP Addresses

- First, an inventory of your components

Component	State
Apache 2.2	OK
Tomcat 6	OK
MySQL 5	OK
Linux (Kernel 2.16.19)	OK
Jboss 4	OK
JRE 1.6.0	OK

The architecture of our application



We define which components to evaluate and the test to perform

- We define to evaluate IPv6 in the user front and also in internal communication between components
- IPv6 support in the datastore was required
- IPv6 support was required in almost all the input forms
- Application logging
- Performance

Your test inventory

User	Module	Id	Name	IPv6 Priority	Observations	Strategy
End User	Object Management, input data	1	New IP address range	High		PLAN
		2	Update IP range	Medium		EXPL
		3	Review IP Ranges	High		PLAN
Admin	Configuration	4	Configure object	Medium		PLAN
		5	Configure system	Medium		EXPL
	Datastore management	6	Backup	Low		EXPL
		7	Backup	Low		EXPL

An Example of a testing sheet

User	Module	Id	Name	Priority IPv6	Observations	Sessions IPv4	Sessions IPv6
Normal	ROA Management	9	New ROA	**HIGH**	There is processing with the IP. Check parser of IP addresses	Mauricio 20120620	Mauricio 20120622
Normal		10	List ROAs	**MEDIUM**	Check that IP addresses are displayed correctly	Mauricio 20120620	Mauricio 20120622
Normal		11	IP addresses in ROAs	**HIGH**	There is processing with the IP. Check parser of IP addresses. Try several formats valid and invalid	Mauricio 20120620	Mauricio 20120622

Case 1: The random turtle

- Some users reported the app to be very slow … sometimes. We reproduced it in production.
- But testing environment was perfect. We review code, components, etc.
- We found that it was an IPv6 problem hidden by Happy-Eye-Balls (RFC) in some browsers.
- We checked all production components (network, server, apache) and all ok … except that sysadmin forgot to enable v6 in JBOSS
- Tip: Have an ipv4-only hostname (just A record) and an ipv6-only (just AAAA) it would make your troubleshooting easier.

Case 2: The creepy IPv6 Address

- One user reported that the app crashed after introducing their IPv6 address (or prefix)
- After some digging we found that his prefix was something like: 2001:db8:1::1::/48 (Can you spot the error?)
- Neither the input or the core components were validating all the cases of IPv6 syntax
- Tip: Validate all the cases of IPv6 syntax. They are trickier than in IPv4

Case 3: Literals are bad (literally)

- An old third party application that we were testing. Apparently very simple (just web) it did not work at all.
- After some code digging and network sniffing we found literals in the code (IPv4 addresses hardcoded in the app)
- To make the app work in IPv6 it needed some hard work
- Tip: IPv6 literals are bad, much worse IPv4 because even simple apps won't work on some transition technologies (i.e. NAT64)

Conclusions

- Testing software is a good practice, it may not be easy, pretty or cheap. But is it good
- Face it, IPv4 is running out and you need to support IPv6 and better now than late
- If you want to sleep thigh, test your apps with IPv6