Developer: I'm so glad I'm not a Networks Engineer and I don’t have to worry about IPv6!
I'm so glad I'm not a Networks Engineer and I don't have to worry about IPv6!
PREPARING APPS FOR IPV6

A SOFTWARE DEVELOPERS GUIDE TO WRITING AND MIGRATING NETWORKED APPLICATIONS FOR USE ON IPV6 NETWORKS

Authors: Andy Newton; Sofía Silva Berenguer
- **Who am I?** I ask it myself every morning 😊
  - I used to work for LACNIC
  - Doing a Master’s in University Carlos III of Madrid now

- **Why am I here?** I wanted to come to Zurich and needed an excuse 😊
  - IPv4 is dead
  - Network engineers are aware of this
  - But what happens at higher layers? Is it really transparent for them?

- **Who else should be here?**
  - Software architects, developers, engineers, etc.; computer programmers; any other curious person.

- **What is this presentation for?**
  - *Developers*: Checklist of areas to investigate in your software’s code
  - *Users*: Inform yourself about the issues your software may encounter with an IPv6 deployment.
WHAT AM I GOING TO DELIGHT YOU WITH?

- Strategies for apps supporting both IPv4 and IPv6
- Sockets
- Proxy and Application Servers
- Format and Comparison
- Persistence and Databases
- URIs/URLs
- IP Geolocation
- Address Types and Special Addresses
- Infrastructure Connectivity
- DNS Considerations
- Miscellaneous “Gotchas”
STRATEGIES FOR IPV6-READY APPS

- IPv6-only app or add IPv6 support to current app?
- IPv6-only (Multiple versions of your app)
  - Users may find it difficult to know which version to use (IPvWhat?)
  - Maintain two versions of almost the same source code
  - Possible Solutions:
    - Hybrid app
    - Wrapper app -> Will probably be too complex
- So? Probably the best option is add IPv6 support to the current app and have an **hybrid app**.
- Or… become a chef and forget about all this 😊
SCENARIOS

- **Best-case scenario** -> IPv4/IPv6 app on dual-stack platform
- **Not-so-good scenario** -> IPv6-only app on dual-stack
- **Another scenario** -> IPv4/IPv6 app in IPv4-only system

- Check ‘RFC 4038 - Application Aspects of IPv6 Transition’
SCENARIOS (CONT.)

- IPv6-only app on dual-stack device
  - App will be IPv6-only and won’t work in IPv4-only platforms
  - *IPv4-mapped* IPv6 addresses (::FFFF:x.y.z.w) would allow app to interoperate with both IPv4 and IPv6 nodes
  - **Be careful!!** Internal IPv4-mapped addresses may be disabled by default due to security concerns (bypass of IPv4 filters)

- IPv4/IPv6 app in IPv4-only system
  - App has to handle the case of IPv6 not being available
SOCKETS

- Changes to app will depend on the programming language being used

- Two categories for changes:
  - 1) hostname lookups
  - 2) generalization of the socket calls to accommodate both IPv4 and IPv6

- Check if calls are IP version-agnostic

- Change AF_INET to AF_INET6 if necessary

- May need to change bind address to "::"

- More details in Guide¹
PROXY AND APPLICATION SERVERS

- Socket mgmt may be delegated to proxy and application server software and frameworks.
- **BUT** server framework has to be set up to listen on an IPv6 port (simple task but don’t forget to do it!)
- Verify IPv6 support was specified during compilation. E.g. PHP and nginx
- JVM-based servers -> runtime property “java.net.preferIPv4Stack” FALSE by default
  - **BUT** many application servers such as JBoss AS / WildFly set the value to TRUE in their startup scripts.
Leading zeros can be omitted both in IPv4 and in IPv6

**BUT** three different formats for IPv6 addresses (full, compressed, and IPv4-embedded)

*And it gets better!* An IPv6 address can be compressed and also have an embedded IPv4 address, and the hexadecimal characters can be either in upper or lower case

So… string comparisons are practically impossible in IPv6 😞

*Regular expressions*: simple for IPv4 **but** very bad performance for IPv6

*Numeric expressions*: IPv4 addresses -> unsigned 32-bit integer (optimizes both storage and comparison) **but** many platforms do not have a 128-bit integer data type
So? What can we do?
- Lower 64 bits of IPv6 addresses -> end-site
- Upper 64 bits -> to route Internet packets to and from the end-site.
- => Depending on how the IP address is used, the application may drop part of the address.

BUT it could get even more interesting!
- 64-bit numeric value defined as a floating-point data type
- Most of which offer 57 bits of precision
- Possible ‘hack’: Analyse current public IPv6 space issued by IANA to RIRs
  - Addresses start with: 2001, 2002, 2003, 2400, 2600, 2610, 2620, 2800, 2A00 and 2C00
  - 10 bit patterns that could be enumerated into a 4-bit field
  - Reduction from 16 to 4 bits, 64-bits prefixes could be represented with only 52 bits
FORMAT AND COMPARISON (CONT.)

- BCP -> use tailored library
  - IPAddr in Ruby; IPy or built-in ipaddress in Python; Net::IP and NetAddr::IP in Perl, PEAR’s Net_IPV4 and Net_IPV6 classes in PHP; built-in InetAddress classes in Java

These libraries also resolve issues surrounding textual presentation and textual validation that are important with UIs.

- **IETF guidance** regarding the presentation of IPv6 addresses in text form\(^2\)
  1. Do not display leading zeros
  2. Use the :: notation when possible but only on the last series of zero bytes and never for just one
  3. Lowercase the hexadecimal characters
FORMAT AND COMPARISON (CONT.)

- Length of IPv6 addresses is unpredictable.
- Some could be too long to be included in messages.
  - Non-essential parts could be hidden
- User input of IP addresses
  - Do not use separate text boxes
  - Offer one text area -> Users ♡ cut & paste
PERSISTENCE AND DATABASES

- For logging source IP addresses, tracking user logins, creation of whitelists, etc.
- How much storage space is needed to persist IPv6 addresses?
- *Strings*: `VARCHAR(45)` instead of `VARCHAR(15)`
- *Binary*: 16 bytes instead of 4 bytes
- MySQL
  - `BIGINT` is not enough but could be used if part of the IP address can be dropped
  - Use *Spatial extensions* to determine if an IP address falls within a certain range (as B-tree indexes do not perform well)
    - IP networks are modelled as rectangles and R-tree indexes are used
      (Much better performance 😊 Yay!)
  - `INET6_ATON` and `INET6_NTOA` instead of `INET_ATON` and `INET_NTOA` functions
PERSISTENCE AND DATABASES (CONT.)

- **PostgreSQL**
  - Since PostgreSQL 7.4 (2003) -> built-in datatypes *inet* and *cidr*, both capable of representing IPv6 addresses.
  - *IP4r* built extension, which leverages PostgreSQL's GiST extensible indexing feature (also uses R-Tree indexes)

- **Oracle**
  - NUMBER data type unusable for IPv6 (it can represent at most 38 significant digits)
  - Also has GIS extensions but they use NUMBER as the base type
  - Solutions:
    1. Store them as a string of characters
    2. Drop some of the lower-order bits and represent high-order bits by NUMBER
URIS / URLs

- E.g., for RESTful communication to server-based applications

- **Basic syntax of all URLs:** `[scheme]://[host]:[port]/[path][query]`

- **In IPv6:** Need to escape `':'` by square brackets (`['` and `']`)

  https://[2001:500:4:13::125]:443/

- **Care needs to be taken when constructing URLs!!**
  - IPv6 may cause errors in the host portion of the URI.
  - Code may need to check if the host is an IPv6 address and, if so, escape it.
  - (you can find examples in the Guide¹)
IP GEOLOCATION

- Geolocation of client based on client’s IP address
- Depending on **IP geolocation database**, IPv6 may not be supported
ADDRESS TYPES AND SPECIAL ADDRESSES

- **Loopback** address (localhost) -> to connect to other processes on the same node (IP sockets)
  - IPv4 -> between 127.0.0.0 and 127.255.255.255 (usually 127.0.0.1)
  - IPv6 -> only one loopback address, ::1

- Clients should resolve the “localhost” hostname instead of using an IP address.

- Servers should use IP address libraries if available
  - Ruby, Java, Python and other languages, are dependent on the underlying operating system to map “localhost” to a loopback address (‘localhost6’ in some OSs)
IPv6 has **multicast** instead of broadcast

Portable code -> use multicast both for IPv4 and for IPv6

All nodes on a local network -> 224.0.0.1 and ff02::1 (no portable way to address both IPv4 and IPv6 multicast nodes with a single address)

Dual-stack nodes will receive duplicated data so tags should be used

Check *IANA IPv6 Multicast Address Registry*[^3] to determine appropriate IPv6 multicast address when porting functionality from IPv4

[^3]: http://www.iana.org/assignments/multicast-addresses
IPv4-compatible IPv6 addresses (::192.168.0.1/96) (DEPRECATED) and IPv4-mapped IPv6 addresses (::ffff:192.168.0.1/96)

Be careful!
- Multiple representations possible
- Different behaviours for different languages.

Link-local addresses -> only valid in the local network

Be careful!
- Link-local addresses are not routable
INFRASTRUCTURE CONNECTIVITY

- C-S and S-S connections must be considered

- **Enable IPv6 in the OS** -> Not always enabled by default
  - Set up interfaces (use static addresses!!)
  - How? Highly dependent on OS/distribution
  - **Be careful** with Windows servers! Microsoft’s Teredo service has been deprecated.

- **Firewalls/Filtering**
  - Probably two or more levels of filtering (local, network firewall, border firewall)
  - Allow traffic to/from app port (both in IPv4 **and** in IPv6)
  - ICMPv6 is **essential** for IPv6 to work properly (E.g. ND) and for troubleshooting (E.g. Echo request/reply) (See RFC 4890)
  - Allow traffic to/from multicast and link-local addresses
  - **Recommendation**: flexible local firewall; more filters at network/border firewall
DNS CONSIDERATIONS

- Protocol used to carry DNS query vs protocol about which we are asking
- If app will be accessed remotely -> add (or ask sysadmin to add) IPv6 info to app server name
  - AAAA to translate name into IPv6 address
  - A single hostname can have A records and AAAA records
- If app will access other services by name -> make sure it queries both A and AAAA records
  - Being able to translate a hostname into an IPv6 address doesn’t mean that the server application supports IPv6 -> client app may fail to establish connection
  - Solution: client app should request all IP addresses and try until a working address is found (default version or runtime decision) (E.g. Happy eyeballs)
When translating names <-> addresses:

- Make sure you use functions that support IPv6
- Set preference (prefer IPv4 or IPv6?)
MISCELLANEOUS ‘GOTCHAS’

- **Apps that rely on CDNs**
  - Does the CDN serve content over IPv6? If it doesn’t how will the content be proxied between its IPv4 servers and your IPv6 customer?
  - Have AAAA DNS records been provisioned for serving CDN data to IPv6 clients?

- **Apps that publish **APIs** or use APIs in which an API call takes an IP address as a parameter**
  - Check that the parameter can be passed as either an IPv4 address or an IPv6 address
Virtual Machines and System Containers and Clouds

- Check whether host and any virtual machines hosted within support IPv6

The Obvious But Overlooked

- Config files and other config data
- Monitoring and alert systems
- Log analyzers and analytic engines

Documentation

- Include IPv6 addresses in documentation for software (Use prefix 2001:db8::/32)
Supporting IPv6-only Networks

May 4, 2016

At WWDC 2015 we announced the transition to IPv6-only network services in iOS 9. Starting June 1, 2016 all apps submitted to the App Store must support IPv6-only networking. Most apps will not require any changes because IPv6 is already supported by NSURLSession and CFNetwork APIs.

If your app uses IPv4-specific APIs or hard-coded IP addresses, you will need to make some changes. Learn how to ensure compatibility by reading Supporting IPv6 DNS64/NAT64 Networks and watch Your App and Next Generation Networks.

↩ Back to News
REFERENCES

3. http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml
   - http://www.internetsociety.org/deploy360/start/
KEEP CALM AND DEPLOY IPv6
No questions? GREAT!! I’m glad I was so clear!