IPv6 programming for Erlang/OTP

Kenji Rikitake
ACCMS/IIMC, Kyoto University
30-MAR-2012
Twitter: @kenji_rikitake
kenji.rikitake@acm.org

Contents

Trying IPv6 on Erlang/OTP is **EASY**

(Very brief) introduction to IPv6

Erlang handling of IPv6 addresses

Erlang/OTP TCP/IP architecture

IPv6 application examples

IPv6 programming pitfalls

Bugs and issues on R15B

Trying IPv6 on Erlang is **EASY**

R15B can handle IPv6 services

- Address format is the (only) major difference

It's ready on major operating systems

- Linux, FreeBSD, Windows 7, etc.

Try free tunneling services for testing

- Enabling IPv6 connectivity over IPv4
- Hurricane Electric's Tunnel Broker

  http://www.tunnelbroker.net/

What is IPv6?

Internet Protocol version 6

- IETF recommendation: July 1994 as "IPng"
- Code base stabilized by 2006 (KAME Project)

Address space: core difference from IPv4

- IPv4: 32 bits -> IPv6: 128 bits
- IPv4 address blocks have been used up

  IANA pool exhausted on 3-FEB-2011

Large-scale apps should migrate to IPv6

- New users may only be able to use IPv6

How IPv6 works (1)

Unicast address assignment in bits

- Network part: 64, Host part: 64
  - Address aggregation occurs to consolidate the routes
- Global ID (48) + Subnet (16) + Host (64)

Host ID: automatically generated or managed

- Stateless autoconfiguration for each interface
  - Host IDs derived from the hardware address
  - Required for boot time neighbor discovery
- Stateful configuration, through DHCPv6
- Host ID can be randomized to enhance privacy

How IPv6 works(2)

Addresses: eight 16-bit hex numbers

`2001:db8:cafe:babe:face:b00c:1234:5678`

Netmasks: usually /64, variable (as CIDR)

Consecutive zeros abbreviated as "::"

`2001:db8:cafe:babe::/64` <- network

`::1 = 0:0:0:0:0:0:0:1` ("localhost")

`2001:db8::1 = 2001:db8:0:0:0:0:0:1`

On URL: use brackets (RFC5952, RFC3986)

`http://[2001:db8:2::50]:80/index.html`

Reverse-lookup zone format: split by hex digits

`e.b.a.b.e.f.a.c.8.b.d.0.1.0.0.2.ip6.arpa`

`See my v6hex module for handling the hex digits`

What is IPv6? (3)

Extensive use of multicasting

Multicast addresses (ff00::/8) have scopes

- interface/machine-local (e.g., ff01::1)
- link/subnet-local (e.g., ff02::1)

  Equivalent to link-level broadcast

Neighbor Discovery Protocol (NDP)

- Solicitation/advertisement of routers/hosts

  Equivalent to ethernet ARP, a part of ICMPv6

What is IPv6? (4)

Routers no longer make packet fragments

- Host-to-host path MTU discovery needed

  Finding out the maximum length of IP packet which will be transferred **without fragmentation**

- Packets exceeding MTU will be discarded

  ICMPv6: Packet Too Big message

- Minimum MTU: 1280 bytes

  Exchanging large UDP packets will be affected

  OS protocol stacks will negotiate the MTU, but end-point programs may also need to be aware of Path MTU

IPv4-mapped IPv6 addresses

Showing IPv4 nodes in IPv6 addresses

- Uses address space of ::ffff:0:0/96
- IPv4: 192.168.0.1 = IPv6 ::ffff:192.168.0.1
  - That's ::ffff:c0a8:1 (in pure hex notation)
  - See RFC4291 Section 2.5.5.2

Interpretation is solely OS-dependent

- IPv4-mapped address used in the source part means the connection comes from an IPv4 node
- Some OS disables this by default
  - Allowing pure IPv6 connection only for IPv6 sockets
  - FreeBSD: net.inet6.ip6.v6only = 1 (disabled)
  - See RFC3493 Section 5.3

Erlang/OTP IPv6 address format

8-element tuple of 16-bit unsigned integers

From R15B lib/kernel/src/inet.erl:

```
-type ip4_address() ::
  {0..255,0..255,0..255,0..255}.
-type ip6_address() ::
  {0..65535,0..65535,0..65535,0..65535,
    0..65535,0..65535,0..65535,0..65535}.
-type address_family() :: 'inet' | 'inet6'.
inet_parse:address(
    "2001:db8:cafe:babe:face:b00c:1234:5678").
› {ok,{8193,3512,51966,47806,64206,45068,4660,
  22136}}
```

Tip: Erlang can handle hex numbers

Adding **16#** prefix to hex numbers will ease coding IPv6 address with Erlang tuples

```
6> {ok, A1} =
  inet_parse:address("2001:db8:cafe:babe::1").
{ok,{8193,3512,51966,47806,0,0,0,1}}
8> A2 = {16#2001, 16#db8, 16#cafe, 16#babe,
  16#0, 16#0, 16#0, 16#1}.
{8193,3512,51966,47806,0,0,0,1}
9> A1 =:= A2.
true
```

(Thanks to Fred Hébert for telling me about this idea!)

Erlang/OTP TCP/IP architecture

User application modules and programs	**Written in Erlang**
kernel gen_tcp, gen_udp, gen_sctp modules (TCP/UDP/SCTP socket interfaces)	
kernel inet_*, inet6_* modules (lower-level access to TCP/IP functions)	
erts/preloaded/src/prim_inet.erl (interface to the linked-in drivers)	
erts/emulator/drivers/common/inet_drv.c (tcp_inet/udp_inet/sctp_inet linked-in drivers)	Linked-in drivers (C code)
OS protocol stack (system calls, socket-related libraries)	OS kernel and libraries

TCP/UDP/SCTP code needs little mods

Erlang/OTP network code is highly abstract

OTP library firmly distinguishes between IPv4 and IPv6 address families

e.g., kernel/src/**inet**_tcp.erl .vs. **inet6**_tcp.erl

- inet or **inet6** address family info required
- connect/{3,4} and listen/2 functions accept the inet6 option in gen_tcp and ssl modules

And that's (almost) all you need to do

- Note: the address family option must match with the IP address passed on to the function

How to determine if IPv6 is supported

Deciding by "localhost" is resolvable to "::1"

inet:getaddr/2 looks up the DNS and returns the address of specified family (inet/inet6)

```
% from Mochiweb mochiweb_socket_server module
ipv6_supported() ->
  case (
   catch inet:getaddr("localhost", inet6)) of
      {ok, _Addr} -> true;
      {error, _} -> false
  end.
```

gen_tcp:connect/3 Address parameter

You only have to pass on the address tuple

If Address is a hostname:

- tcp module name in ERL_INETRC is effective

  To change this for IPv6, add the following line:

  ```
  {tcp, inet6_tcp}. % default: inet_tcp
  % Don't forget the ending period
  ```

If Address is a tuple:

- Choose the family by BIF tuple_size(Address)

  4 -> IPv4 (inet_tcp), 8 -> IPv6 (inet6_tcp)

Same behavior on gen_udp and gen_sctp

More OTP IPv6-compatible functions

inet_parse:address/1 (address string -> tuple)

inet_parse:ntoa/1 (tuple -> address string)

inet:getaddrs/2 (2nd arg: address family)

inet:gethostbyaddr/1 (tuple -> hostent)

inet_res:gethostbyaddr/1 (DNS backend)

inet_res:gethostbyname/1 (DNS backend)

- inet_res resolvers will try to return IPv6 address first when the following line is set in ERL_INETRC (and IPv4-mapped IPv6 address for IPv4 addresses):

```
{inet6, true}. % default: false
% Don't forget to include the period!
```

IPv6 support on Erlang programs

"grep **inet6**" helps to look up the source code

TCP-based Web servers are OK

- Mochiweb, Yaws (including SSL/TLS)

TCP/UDP network programs are also OK

- Tsung, ejabberd

Rewriting needed for those handle ICMPv6

- Procket (socket tweaking tool)

  ICMPv6 (protocol 58) =/= ICMPv4 (protocol 1)

  See my (experimental) example fork on GitHub

How to choose IPv4 or IPv6

Web/TCP servers: use multiple instances

- Use at least one for each protocol

DNS: preference strategy required

- RFC3484 recommends IPv6 first, then IPv4
- Reality: very few sites support IPv6 yet
- A simple workaround example
  - Look up AAAA RR first with timeout (~200ms)
    - If found, then use the IPv6 address for access
  - If not found, look up A RR (falling back to IPv4)
  - Example code in my v6hex:v64adrs/{1,2}

Bugs and issues on R15B

Distributed Erlang on IPv6 doesn't work

- `-proto_dist inet6_tcp`
- `epmd` doesn't listen on the IPv6 port
  - Patch exists but not accepted by OTP team
- Multiple daemons for multiple transports?

Interface identifiers (IIDs) not supported

- Interface name after '%' e.g., ff02::1%em0
- Required for link-scoped multicast addresses

ICMP and raw sockets (aka black magic)

Acknowledgments to:

People helping the code development

- Francesco Cesarini

  Who suggested me to give this talk

- Michael Santos (the author of procket)
- Frédéric Trottier-Hébert (for 16# prefix)

and all the participants of EF SF Bay 2012!

References (1)

- v6hex: https://github.com/jj1bdx/v6hex
- Mochiweb: https://github.com/mochi/mochiweb
- Procket: https://github.com/msantos/procket
  - mine with ICMPv6: https://github.com/jj1bdx/procket
- Erlang/OTP documentation
  - Inet configuration, ERTS User's Guide
  - inet module, kernel reference manual
- Erlang/OTP source code

lib/kernel/src/inet*.erl

(Read the files many times to understand the details)

References (2)

- Kevin R. Fall, and W. Richard Stevens, TCP/IP Illustrated, Volume 1, **Second Edition**: The Protocols, Addison-Wesley, 2012, ISBN 9780321336316 (including full IPv6 explanation)
- W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff, UNIX Network Programming, Volume 1, **Third Edition**: The Sockets Networking API, Addison-Wesley, 2004, ISBN 978013141155 (describing basic coding techniques)
- Dan York, Migrating Applications to IPv6, O'Reilly, 2011, ISBN 9781449307875 (recommended as in introductory reading)