SEEBURGER
BUSINESS INTEGRATION

Java-Anwendungen für IPv6 fit machen

IPv**6** Kongress 2013, 6.+7. Juni Frankfurt/Main

b.eckenfels@seeburger.de

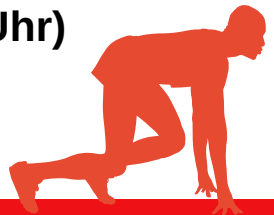
Java-Anwendungen für IPv6 fit machen

Bernd Eckenfels (Chief Architect, SEEBURGER AG)

Dass die Java-Plattform IPv6 unterstützt, ist allgemein bekannt – oder es wird vermutet. Aber sind dazu Änderungen an Anwendungen notwendig, die bisher nur für IPv4 getestet wurden? Welche Funktionen der IPv6-Protokollfamilie werden unterstützt? Welche Besonderheiten sind zu beachten, um Dual-Stack-fähige Anwendungen in Java zu erstellen?

Der Vortrag betrachtet die Java-API für Netzwerkkommunikation und untersucht diese auf Relevanz für IPv6. Ein besonderes Augenmerk wird auf die Umsetzung von Dual-Stack-fähigen Geschäftsanwendungen (TCP- und TLS-Protokoll) gelegt. Implementierungsdetails von Java SE 7 (OpenJDK, Oracle und IBM) sind Teil des Vortrags.

(07.06.2013 - 14:00-14:30 Uhr)



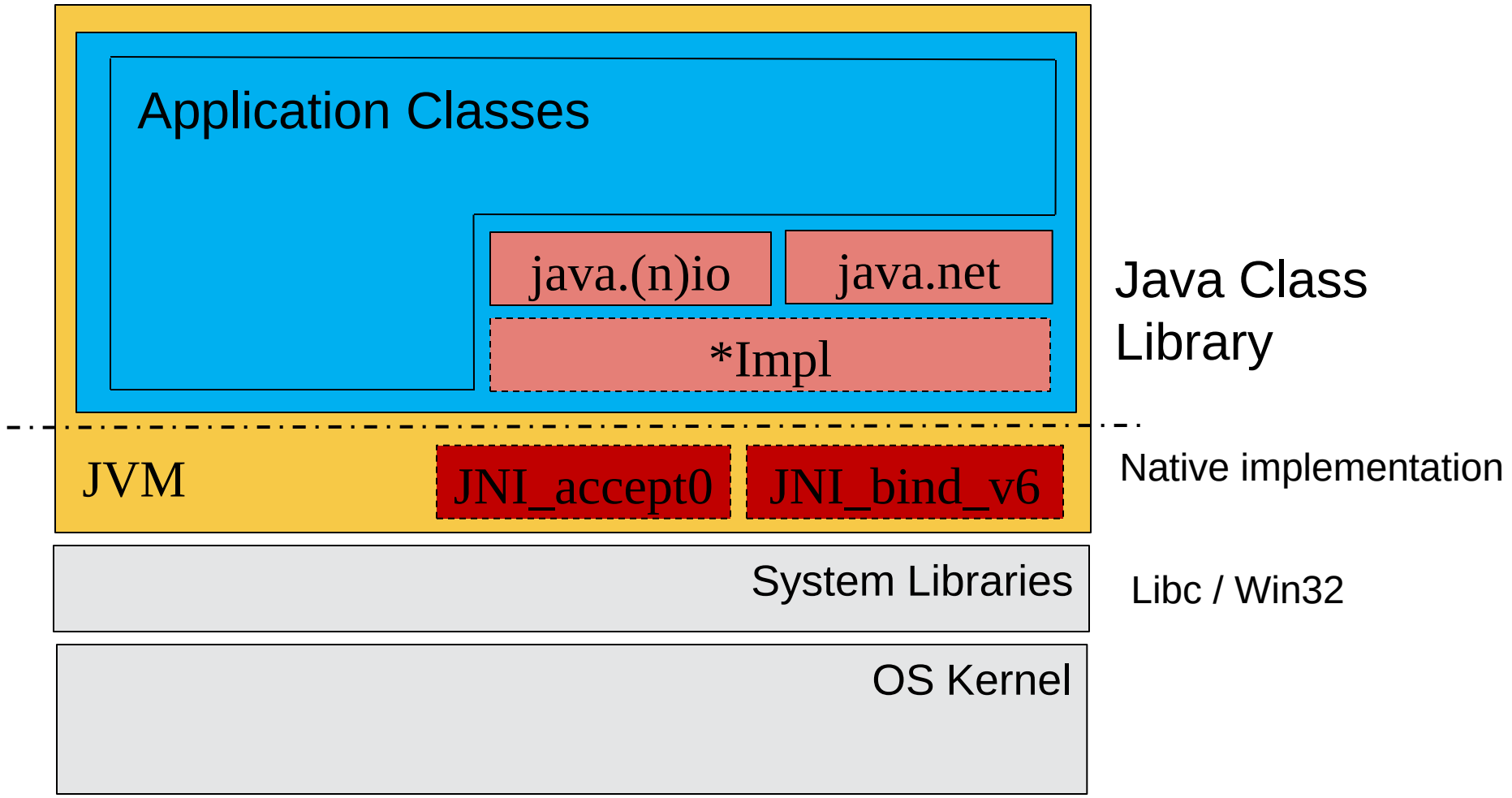
Themenübersicht

- Java Networking Architektur
- Code Samples
- Stolpersteine für Anwendungen
- Dual-Stack Herausforderungen
- Unterstützung von IPv6 Spezialitäten

IPv6 mit Java – Eine Zeitlinie

- 2002: Java 1.4 veröffentlicht mit initialem Support für IPv6 auf Solaris und Linux
- 2002: Windows XP SP1 mit IPv6 support (keine Dual Stack Socket)
- 2004: Java 5 - Spezielle Two-Stack Unterstützung für Windows Plattform
- 2006: Java 6 - Keine zusätzlichen IPv6 Features
  - Einige Linux Distributionen liefern IPV6ONLY=0 aus
- 2007: Windows Vista mit echten Dual-Stack Sockets
- 2011: Java 7
  - Unterstützung für Dual Stack Sockets unter Windows Vista ff.
  - V6ONLY=0 als Default
  - Socket completion (NIO.2)
  - Multicast Listener Discovery v2 (RFC3810)

2014: Java 8
Keine Neuerungen für IPv6 geplant



java.net

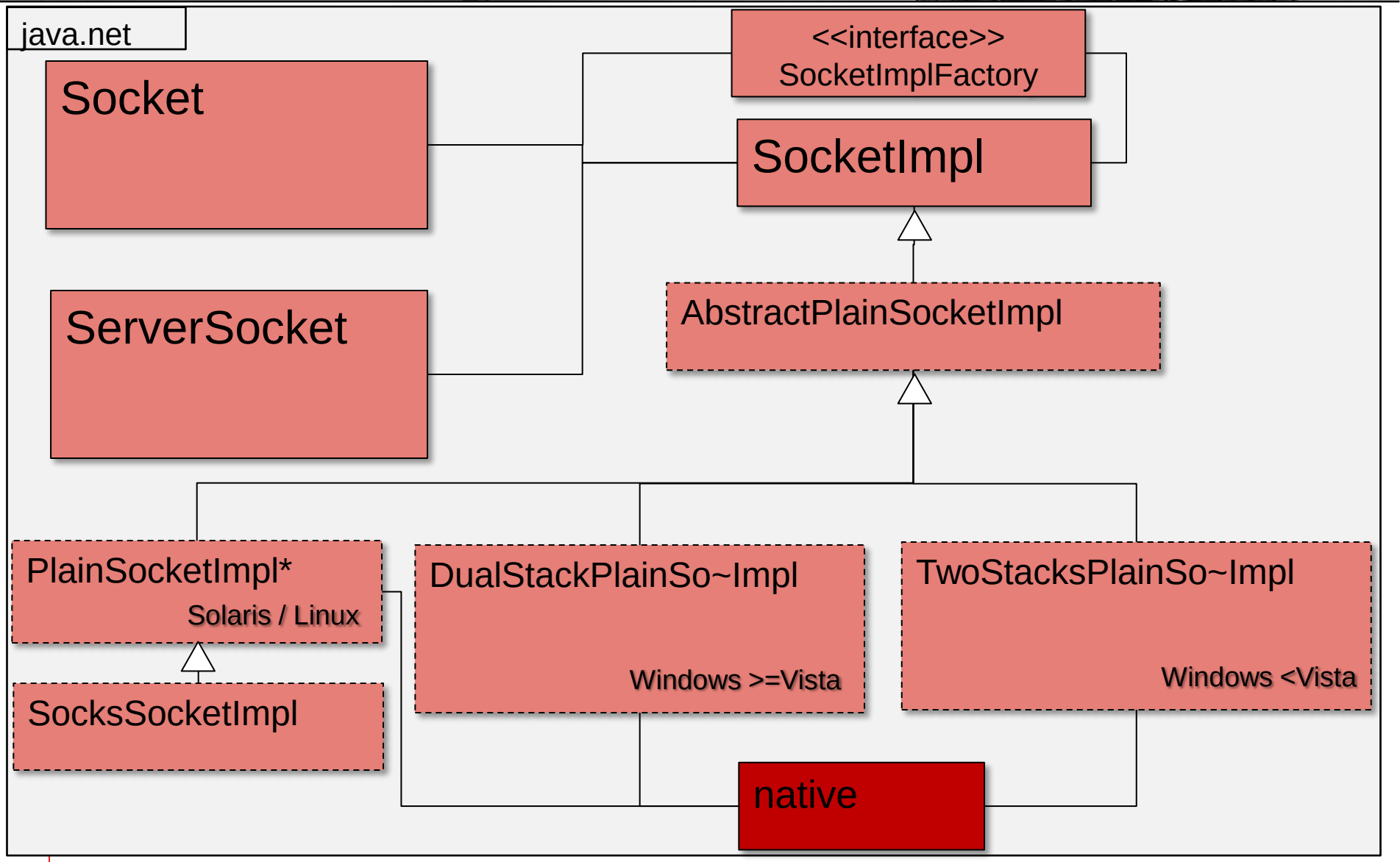
InetSocketAddress
- address : InetAddress
- port : int

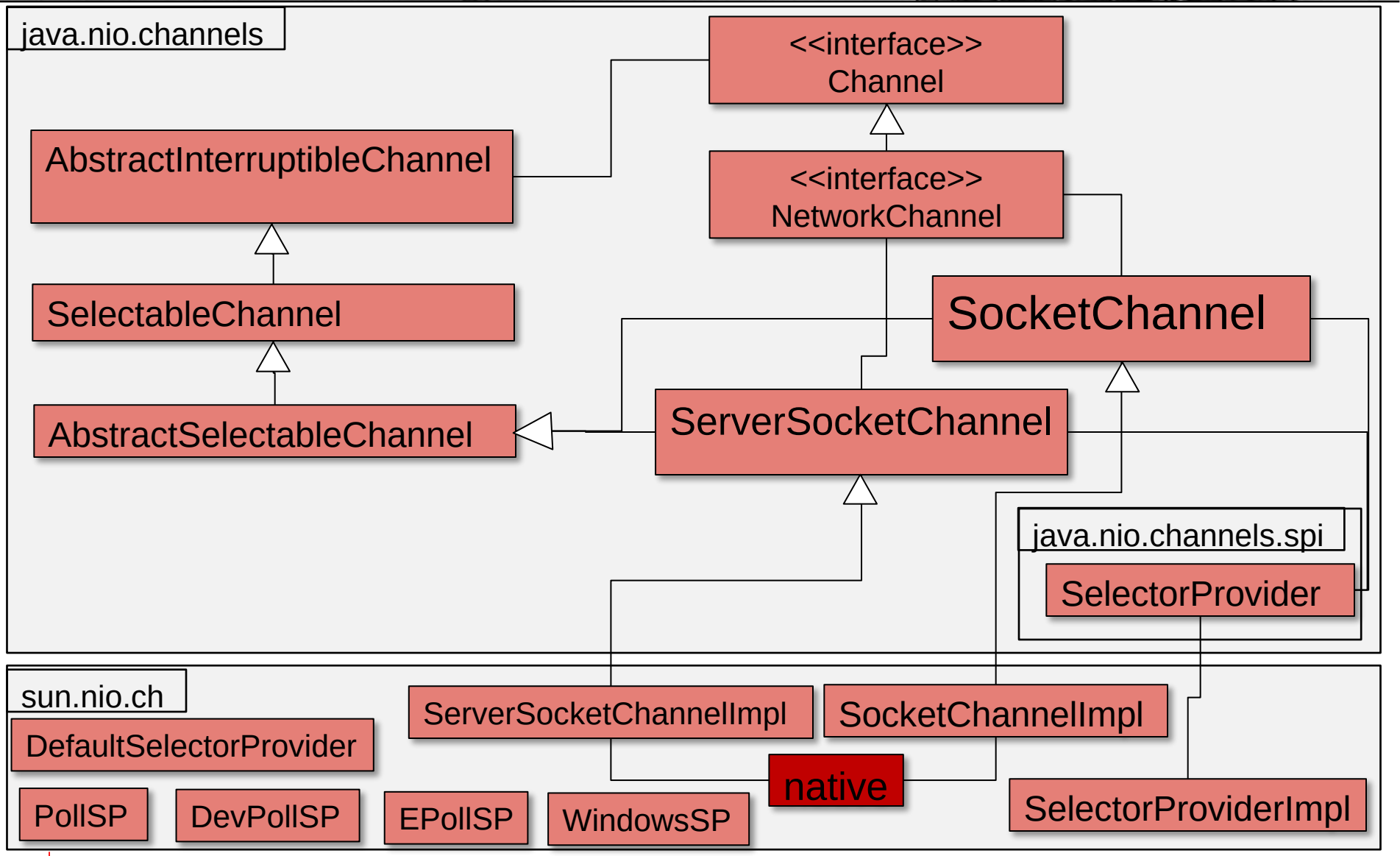
InetAddress
- getAllByName()
- getByName()
- getByAddress()
- isMulticastAddress()
- isReachable()

Cache
Permissions
Nameservice

Inet4Address

Inet6Address
- getByAddress()
- getScopeId()
- isIPv4Compatible()





Sample: Address Family neutral resolving

```
package net.eckenfels.ipv6con;

import java.net.*;
import static java.lang.System.*;

public class Sample1 {
    public static void main(String[] args) throws UnknownHostException {
        InetAddress address = InetAddress.getByName(args[0]);
        out.printf("Looking up %s%n -> %s%n    %s%n",
                   args[0], address, address.getClass());
    }
}
```

```
C:\IPv6Con> set CP=-cp target\classes\
C:\IPv6Con> java %CP% net.eckenfels.ipv6con.Sample1 www.kame.net
Looking up www.kame.net
 -> www.kame.net/203.178.141.194
    class java.net.Inet4Address

C:\IPv6Con> java %CP% -Djava.net.preferIPv6Addresses=true ^
                net.eckenfels.ipv6con.Sample1 www.kame.net
Looking up www.kame.net
 -> www.kame.net/2001:200:dff:fff1:216:3eff:feb1:44d7
    class java.net.Inet6Address
```

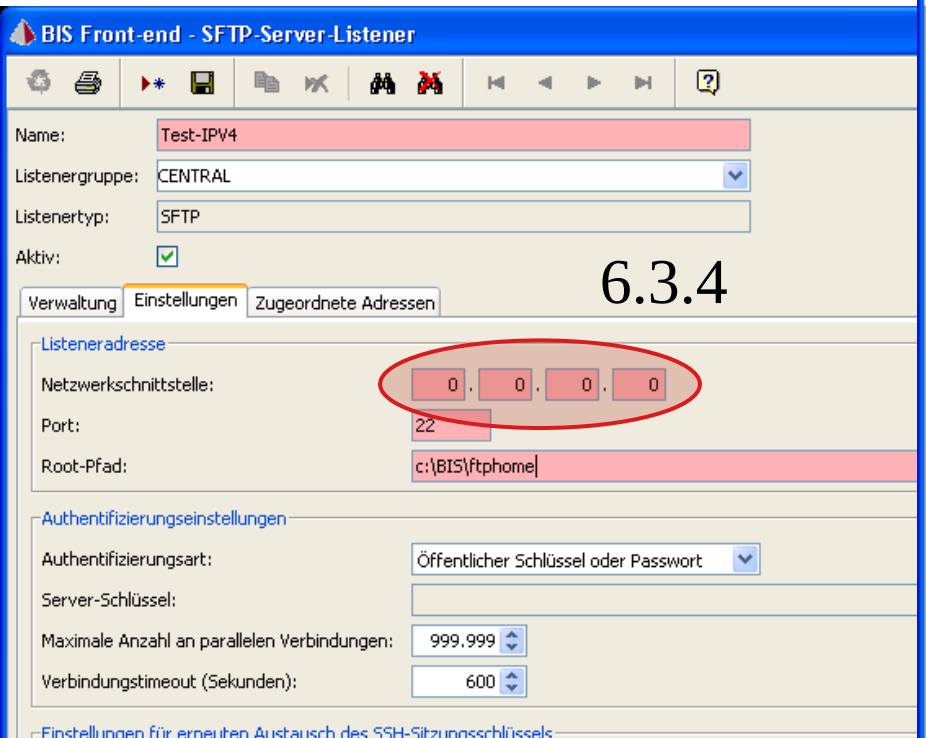
Sample: Address Family neutral parsing

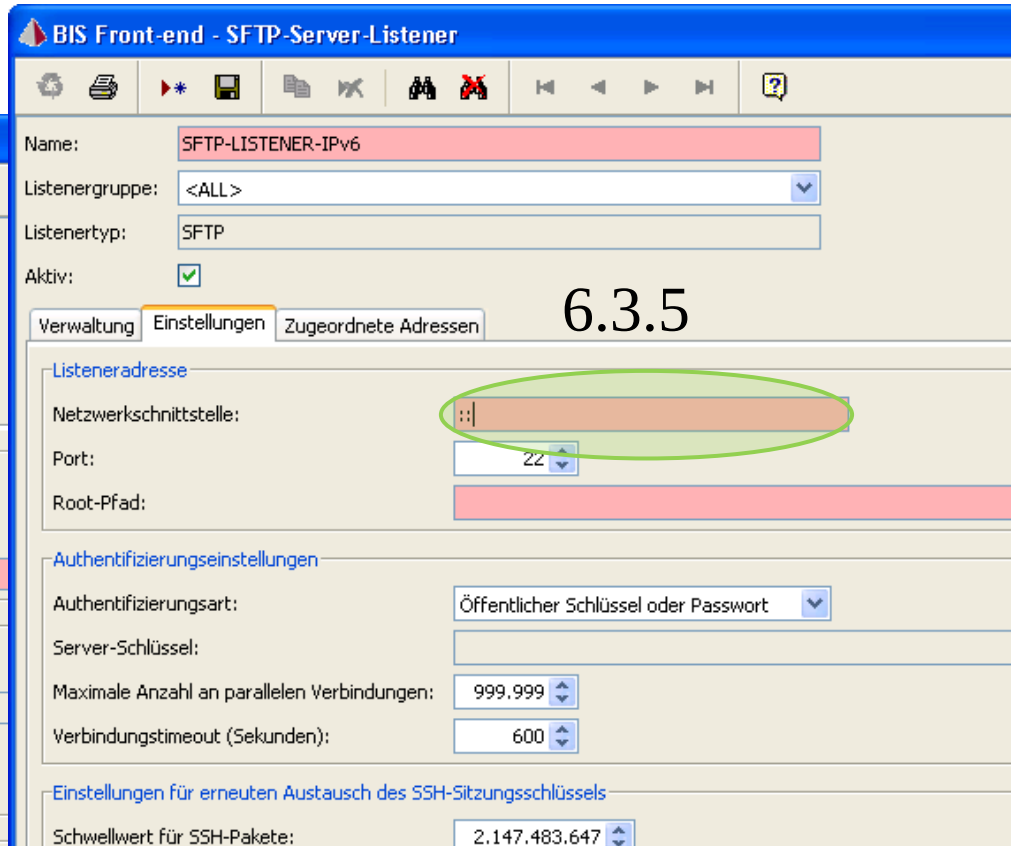
```
package net.eckenfels.ipv6con;

import java.net.*;
import static java.lang.System.*;

public class Sample2 {
    public static void main(String[] args) throws UnknownHostException {
        InetAddress address = InetAddress.getByName(args[0]);
        out.printf("Looking up %s -> %s %s isLoop=%s%n",
                   args[0], address, address.getClass(), address.isLoopbackAddress());
    }
}
```

```
C:\IPv6Con> java %CP% net.eckenfels.ipv6con.Sample2 ::1
Looking up ::1 -> /0:0:0:0:0:0:0:1 class java.net.Inet6Address isLoop=true

C:\IPv6Con> java %CP% net.eckenfels.ipv6con.Sample2 127.0.0.1
Looking up 127.0.0.1 -> /127.0.0.1 class java.net.Inet4Address isLoop=true

C:\IPv6Con> java %CP% net.eckenfels.ipv6con.Sample2 192.0.2.16
Looking up 192.0.2.16 -> /192.0.2.16 class java.net.Inet4Address isLoop=false

C:\IPv6Con> java %CP% net.eckenfels.ipv6con.Sample2 localhost
Looking up localhost -> localhost/127.0.0.1 class java.net.Inet4Address isLoop=true
```

Anwendung fit machen?

- Bei Eingabefelder für Hostnamen keine Änderungen notwendig
- Address Felder (UI+DB) müssen genügend Raum bieten und Zeichen (: und %) zulassen
- Parsen von Konfigurationsdateien oder schreiben von Files beachten
- Protokolle mit eingebundenen Addressen





Anwendung fit machen?

- Kein Support in der JCL für das Parsen von Netzangaben (prefix/prefixlen Notation)
- java.net.URL kann inzwischen RFC2732 (Format for literal IPv6) parsen:

```
public class Sample3 {
    public static void main(String[] args) throws MalformedURLException {
        URL url1 = new URL("http", "2001:DB8::1234:ABCD%13", 80, "");
        out.printf("URL1 toString=%s host=%s port=%d%n",
                   url1.toExternalForm(), url1.getHost(), url1.getPort());
        URL url2 = new URL("http://[2001:DB8::1234:ABCD%13]/test");
        out.printf("URL2 toString=%s host=%s port=%d%n",
                   url2.toExternalForm(), url2.getHost(), url2.getPort());
    }
}
```

```
C:\IPv6Con> java %CP% net.eckenfels.ipv6con.Sample3
URL1 toString=http://[2001:DB8::1234:ABCD%13]:80 host=[2001:DB8::1234:ABCD%13] port=80
URL2 toString=https://[2001:DB8::1234:ABCD%13]/test host=[2001:DB8::1234:ABCD%13] port=-1
```

RFC 3484

(Default Address Selection for IPv6)

Hands Up?!

Dual IP Stack Applications

RFC 3484 – Default Source/Destination Address Selection

- Warum? Multihomed, Tunnels, Scope, Renumber, Privacy Extension, Lifetime

Kernel/Anwendungsschnittstelle auf Basis von `getaddrinfo()` (benutzt die 9 Regeln aus RFC 3484 um eine gewichtete Liste zu erstellen). Betriebsysteme erlauben administrativen Eingriff in diese Gewichtung: (`/etc/gai.conf`, `netsh intf ipv6 show prefixpolicy`)

> “A lot of the information used by the sorting algorithm is not easy to come by. It is therefore highly recommended that no program uses home-grown implementations of `getaddrinfo()`. Just use the system implementation.” (Ulrich Drepper, 2007, http://www.akkadia.org/drepper/linux-rfc3484.html)

Dual IP Stack Applications

- `InetAddress.getAllByName(String)` unterstützt austauschbare Provider
  - Der (Sun) default benutzt `getaddrinfo()` allerdings werden im Ergebnis alle IPv6 Adressen entweder an den Anfang oder das Ende der Liste sortiert (java.net.preferIPv6Addresses=true/false).
  - Alternativer Provider (sun.net.spi.nameservice.provider.1=dns,sun) möglich

- Keine Unterstützung für (paralleles) Verbinden mit mehreren Kandidaten (“Happy Eyeballs” Verfahren).

- Der Java IPv6 Guide vergleicht C-code mit Schleife mit Java Quelltext der nur die erste Adresse probiert (.Net kennt ein WSAConnectByName)

Dual-Stack - Addresslisten

- `getAllByName()` liefert `getaddrinfo()` Liste mit zusätzlicher Protokollgewichtung

```
public class Sample4 {
    public static void main(String[] args) throws UnknownHostException {
        InetAddress[] addrs = InetAddress.getAllByName(args.length>0?args[0]:null);
        for(InetAddress a : addrs)
            out.printf("- %s %s%n", a.getHostAddress(), a.getCanonicalHostName());
   }
}
```

```
C:\IPv6Con> java %CP% net.eckenfels.ipv6con.Sample4 www.heise.de
- 193.99.144.85 www.heise.de
- 2a02:2e0:3fe:100:0:0:0:7 www.heise.de

C:\IPv6Con> java %CP% -Djava.net.preferIPv6Addresses=true ^
                net.eckenfels.ipv6con.Sample4 www.heise.de
- 2a02:2e0:3fe:100:0:0:0:7 www.heise.de
- 193.99.144.85 www.heise.de

C:\IPv6Con> java %CP% net.eckenfels.ipv6con.Sample4 www.kame.net
- 203.178.141.194 orange.kame.net
- 2001:200:dff:fff1:216:3eff:feb1:44d7 2001:200:dff:fff1:216:3eff:feb1:44d7

C:\IPv6Con> java %CP% -Djava.net.preferIPv4Stack=true net.eckenfels.ipv6con.Sample4 www.kame.net
- 203.178.141.194 orange.kame.net

C:\IPv6Con> java %CP% net.eckenfels.ipv6con.Sample4 ipv6.google.com
- 2a00:1450:4016:802:0:0:0:1013 muc03s07-in-x13.1e100.net
```

IPv6 Connection

```
public class Sample5 {
    public static void main(String[] args) throws IOException {
         String host = "www.heise.de"; int port = 80;
         InetAddress address = InetAddress.getByName(host);
         InetSocketAddress sockAddr = new InetSocketAddress(address, port);

         printAndClose( new Socket(host, port) );

         printAndClose( new Socket(address, port) );

         Socket s = new Socket(); s.connect(sockAddr);
         printAndClose(s);
    }

    private static void printAndClose(Socket s) {
        out.printf("Connected %s#%d -> %s#%d%n",
            s.getLocalAddress(), s.getLocalPort(), s.getInetAddress(), s.getPort());
        try { s.close(); } catch (Exception ignored) { }
    }
}
```

```
C:\IPv6Con> java %CP% net.eckenfels.ipv6con.Sample5
Connected /10.0.0.24#56850 -> www.heise.de/193.99.144.85#80
Connected /10.0.0.24#56851 -> www.heise.de/193.99.144.85#80
Connected /10.0.0.24#56852 -> www.heise.de/193.99.144.85#80

C:\IPv6Con> java %CP% -Djava.net.preferIPv6Addresses=true net.eckenfels.ipv6con.Sample5
Connected /2001:0:5ef5:79fd:6c:1cde:6a53:c2ba#57258 -> www.heise.de/2a02:2e0:3fe:100:0:0:0:7#80
Connected /2001:0:5ef5:79fd:6c:1cde:6a53:c2ba#57259 -> www.heise.de/2a02:2e0:3fe:100:0:0:0:7#80
Connected /2001:0:5ef5:79fd:6c:1cde:6a53:c2ba#57260 -> www.heise.de/2a02:2e0:3fe:100:0:0:0:7#80
```

IPv6 Server Socket

- ServerSocket auf ANY nimmt per Default beide Addressfamilien an:

```
public class Sample6 {
    public static void main(String[] args) throws IOException {
        //ServerSocket server = new ServerSocket(1234,50,InetAddress.getByName("0.0.0.0"));
        //ServerSocket server = new ServerSocket(1234,50,InetAddress.getByName("::"));
        //ServerSocket server = new ServerSocket(); server.bind(new InetSocketAddress("::", 12345));
         ServerSocket server = new ServerSocket(1234);
         out.printf("Listening on %s%n",  server.getLocalSocketAddress());
         while(true) {
             Socket client = server.accept();
             printAndClose(client);
         }
    }

    private static void printAndClose(Socket s) {
        out.printf("Connected %s#%d <- %s#%d%n",
            s.getLocalAddress(), s.getLocalPort(), s.getInetAddress(), s.getPort());
        try { s.close(); } catch (Exception ignored) { }
    }
}
```

```
C:\IPv6Con> java %CP% net.eckenfels.ipv6con.Sample6
Listening on 0.0.0.0/0.0.0.0:1234
Connected /0:0:0:0:0:0:0:1#1234 -> /0:0:0:0:0:0:0:1#56185
Connected /127.0.0.1#1234 -> /127.0.0.1#56188

C:\IPv6Con> java %CP% -Djava.net.preferIPv6Addresses=true net.eckenfels.ipv6con.Sample6
Listening on ::/0:0:0:0:0:0:0:0:1234
```

Working with Interfaces

```
import static java.lang.System.out;
import java.io.IOException;
import java.net.*;
import java.util.*;

public class Sample7 {
    public static void main(String[] args) throws IOException {
        Enumeration<NetworkInterface> ifs = NetworkInterface.getNetworkInterfaces();
        for(NetworkInterface i: Collections.list(ifs)) {
            List<InterfaceAddress> as = i.getInterfaceAddresses();
            if (as == null || as.size() == 0) continue;
            out.printf("%s %%%d %s mtu:%d%n", i.getName(), i.getIndex(),
                    i.getDisplayName(), i.getMTU());
            for(InterfaceAddress a : as)
                out.printf("  addr %s /%d%n", a.getAddress().getHostAddress(),
                    a.getNetworkPrefixLength());
        }
    }
}
```

```
C:\IPv6Con> java %CP% net.eckenfels.ipv6con.Sample7
lo %1 Software Loopback Interface 1 mtu:-1
  addr 127.0.0.1 /8
  addr 0:0:0:0:0:0:0:1 /128
eth3 %10 Bluetooth-Ger?t (PAN mtu:1500
  addr fe80:0:0:0:a492:4bb3:e0a0:165e%10 /64
net4 %0 Intel(R) Centrino(R) Advanced-N 6200 AGN mtu:1500
  addr 10.0.0.24 /-1
net9 %19 Microsoft-Teredo-Tunneling-Adapter mtu:1280
  addr 2001:0:5ef5:79fd:3044:1a7f:6a53:c2ba /0
  addr fe80:0:0:0:3044:1a7f:6a53:c2ba%19 /32
```

Bug#6512101 Fixed in Java 8

Special Functions

- `setTrafficClass(int)` setzt bei v6 vor dem connect bits im flowinfo feld (ignoriert in Windows). Kann auch zum Setzen der ToS in IPv4 genutzt werden.
- `isReachable(int)` Benutzt ICMPv6 (oder TCP echo port)

```
public class Sample8 {
    public static void main(String[] args) throws IOException {
        InetAddress a = InetAddress.getByName(args[0]);
        Socket s = new Socket();
        s.setTrafficClass(17);
        s.connect(new InetSocketAddress(a, Integer.valueOf(args[1]).intValue()));
        out.printf("connected %s -> %s%n", s.getLocalSocketAddress(), s.getRemoteSocketAddress());
        s.close();

        out.printf("is %s pingable? %s%n", a, a.isReachable(5*1000));
    }
}
```

```
C:\IPv6Con> java %CP% net.eckenfels.ipv6con.Sample8 www.heise.de 80
connected /2001:0:5ef5:79fd:3044:1a7f:6a53:c2ba:56522 -> www.heise.de/2a02:2e0:3fe:100:0:0:0:7:80
is www.heise.de/2a02:2e0:3fe:100:0:0:0:7 pingable? True
```

Fazit

- Java Anwendungen sind mit geringem Aufwand IPv6 fähig zu machen
  - Problemlos funktioniert dies erst ab Java 7
  - Ohne Systemproperty `preferIPV6` wird IPv4 immer bevorzugt
- Plattformübergreifende Unterstützung von IPv6 oder Dual Stack Anwendungen
- Stolpersteine sind UI, Parsing sowie hart-codierte Adressen (oder `localhost` literal)
- Eingeschränkte Unterstützung für fortgeschrittene Features
  - `getaddrinfo()` wird umsortiert
  - `V6ONLY` kann nicht beeinflusst werden
  - Flowlabel kann nicht definiert/abgefragt werden
- Happy Eyeballs muss selbst implementiert werden

Vielen Dank für Ihre Aufmerksamkeit

Bernd Eckenfels
Chief Architect
SEEBURGER AG

http://itblog.eckenfels.net
http://www.seeburger.de

Beispielcode

https://github.com/ecki/IPv6Con
https://github.com/ecki/WinIPv6Test

Quellen

- Java Networking Guides
  http://docs.oracle.com/javase/7/docs/technotes/guides/net/index.html
  http://docs.oracle.com/javase/7/docs/technotes/guides/net/properties.html
- Networking IPv6 User Guide for JDK/JRE 5.0
  http://docs.oracle.com/javase/7/docs/technotes/guides/net/ipv6_guide/index.html
- Alan Bateman: (blog) IPv6 – History of IPv6 in Java
  https://blogs.oracle.com/alanb/entry/ipv6
- Windows API WSAConnectByName
  http://msdn.microsoft.com/en-us/library/windows/desktop/ms741557%28v=vs.85%29.aspx

- RFC 3493 – Basic Socket Interface Extension for IPv6
- RFC 6555 – Happy Eyeballs http://en.wikipedia.org/wiki/Happy_Eyeballs

Java-Anwendungen für IPv6 fit machen

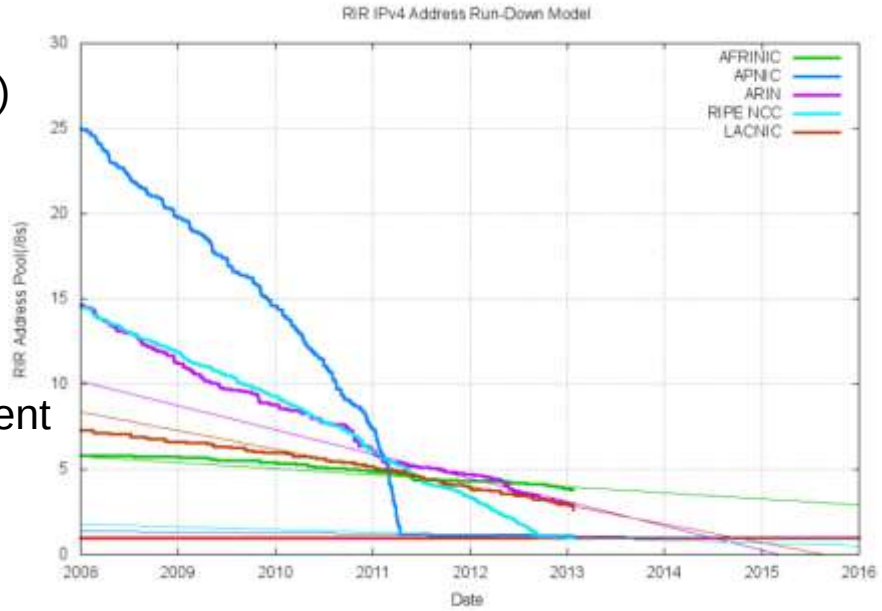
BONUS MATERIAL

IPv6 Prefix Policy and getaddrinfo (Windows native, Teredo)

```
C:\IPv6Test>Debug\IPv6Test.exe www.heise.de
getaddrinfo response 1
        Flags: 0x0
        Address: AF_INET (IPv4) 193.99.144.85
        Canonical name: www.heise.de
getaddrinfo response 2
        Flags: 0x0
        Address: AF_INET6 (IPv6) 2a02:2e0:3fe:100::7

C:\IPv6Test>netsh int ipv6 show prefixpol

Vorgänger   Label  Präfix
----------  -----  ------------------------------
        50      0  ::1/128
        40      1  ::/0
        30      2  2002::/16
        20      3  ::/96
        10      4  ::ffff:0:0/96
         5      5  2001::/32

C:\ IPv6Test>netsh int ipv6 set prefixpol 2001::/32 precedence=40 label=1 store=active

C:\IPv6Test>Debug\IPv6Test.exe www.heise.de
getaddrinfo response 1
        Flags: 0x0
        Address: AF_INET6 (IPv6) 2a02:2e0:3fe:100::7
        Canonical name: www.heise.de
getaddrinfo response 2
        Flags: 0x0
        Address: AF_INET (IPv4) 193.99.144.85
```

A new IP Version? When and How?

- IPv6 was introduced 1998 to solve shortage problem
- IPv6 is coming real soon now:
  - World IPv6 Day/Launch (2011-06/2012-07)
  - Nameserver Infrastructure is ready
    - DENIC since 2004
  - End customer deployment planned
    - (Telekom) this year
  - US Gov mandatory procurement requirement
    - (Federal Aquisition Regulation July 2010)
  - **APNIC RIR has NO IPv4 Addresses left**
    - **Do you need to talk to APAC sites?**



Quelle: http://en.wikipedia.org/wiki/IPv4_address_exhaustion
Geoff Huston's projection of the evolution of the IP pool for each RIR

- How? Dual Stack – a node can have IPv6 and IPv4 addresses

IPv4 Address

IPv4* 192 . 0 . 2 . 16

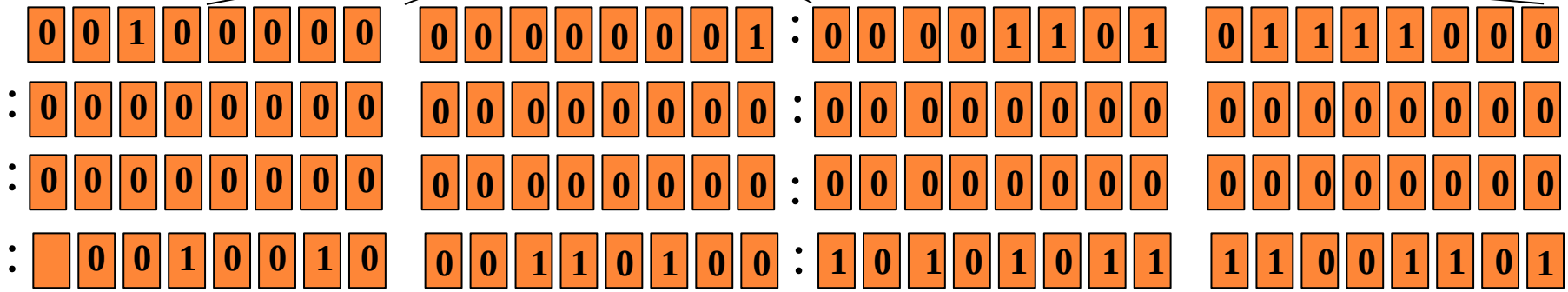
4 bytes/octets = 32bit

2^{32}-1= 4.294.967.295

* RFC5737 IPv4 Address Blocks Reserved for Documentation

IPv6 Addressing Architecture

IPv6** 2001:DB8::1234:ABCD



16bytes(octets) = 128bit

2^{128}-1=340.282.366.920.938.463.463.374.607.431.768.211.455
=3,4*10^{38} = 340 Sextillionen (DEU) = 340 undecilion (ENG)

** RFC 3849 IPv6 Address Prefix Reserved for Documentation

IPv6 Address Notation

1111 1110‘1000 0000‘0..0‘0..0‘... ‘0..0‘0..0‘0000 0001‘0000 0000‘0000 0000‘0000 0001

- 8 groups with 4 hex digits represent 16bit each:

  fe80:0000:0000:0000:0000:0000:0100:0001

- Leading zeros can be ommitted

  fe80:0:0:0:0:0:100:1

- Longest sequence of two or more zero-groups can be replaced with „::“

  fe80::100:1

- In some cases IPv4 notation can be included:

  (fe80::1.0.0.1)

IPv6 Address Architecture and Registry

3+45 bits	16 bits	64 bits
Global routing prefix	subnet ID	Interface ID

- Global routing prefix includes the 3bit prefix „Global Unicast“ 2000::/3 = 001xxxxx….
  - Global routing prefix is assigned by the ISP
- Subnet ID is identifier of link within a site (company) , typical size m=16 bit (/48)
- Interface ID is a 64 bit address on the local network (e.g. build from MAC)

- A node decides based on the routing table where a packet should be sent to
  - Routing table contains only networks (prefix) or default (all others) route

- The network prefix length specifies the number of bits in the IP address which stay the same for all hosts on a network. It is specified as /<number>
  - IANA RIPE DENIC-v6-TINET6 l.de.net
  - 2000::/3 -> 2001:0600::/23 -> 2001:0668:001F:0011::/64 -> 2001:668:1f:11::105

IPv6 Prefixes

Unicast

- 2000::/3 Global Unicast
- fe80::/10 Link local (=169.254.0.0/16)
- fc00::/7 Unique Local Address (ULA) (=10.0.0.0/8, 192.168.0.0/16,…) RFC 4193

Multicast

- ff01::1 All Nodes Interface; ff02::1 All Nodes Link
- All Routers: ff01::2; ff02::2; ff05::2 (Interface, Link, Site)

Special

- ::/128 (all zero) = unspecified
- ::1/128 = loopback address (=127.0.0.1)
- 2001:0:5ef5:79fd:3467:390a:f5ff:ffe6 (Teredo)

Simple Packet Format

- IPv6 RFC 2460 (1998): 39 pages
  - IPv6 Header: 8 fields (size 40byte) + extensions
- IPv4 RFC 791 (1981): 45 pages
  - IPv4 Header: 13 fields (size min 20bytes)

- IP is the packet underlying reliable streams (TCP, SCTP), unreliable datagrams (UDP) and control message (ICMP). In IPv6 TCP and UDP are unchanged, ICMPv6 has new functions.