IPv6-centric application development

Andrew Yourtchenko

@ayourtch

Agenda

- Why IPv6-Centric ?
- What is IPv6 centric ? *IPv6 only clients and servers*
- Languages and OSs.
  - C and POSIX; iOS / OSX; Java; Python; PHP; iOS/OSX, Windows …
- User Interface
- Testing Your Applications: Communities, Lab,…
- Advanced considerations
  - Fragmentation, Addressing

Learning more about IPv6

BRKRST-2616	Addressing Networking challenges with latest Innovations in IPv6
COCIP6-1013	IPv4 Address Exhaustion and IPv6 Progress across Cisco IT
BRKRST-2116	Intermediate - IPv6 from Intro to Intermediate
DevNet-1275	Developing Better Applications with IPv6
BRKRST-2022	IPv6 Routing Protocols Update
BRKSPG-2603	Intermediate - How to Securely Operate an IPv6 Network
LABIPM-2007	Intermediate - IPv6 Hands on Lab
CCSIP6-2006	BMW: Enterprise IPv6 adoption
LABSPG-7122	Advanced IPv6 Routing and services lab
BRKIP6-2100	IPv6-centric application development
BRKRST-2667	How to write an IPv6 Addressing Plan
BRKSPG-2300	Service Provider IPv6 Deployment
PNLCRS-2307	Don't Be Left Behind: Consumer Internet Traffic is Shifting to IPv6, Will your Organization Follow?
BRKRST-2312	Intermediate - IPv6 Planning, Deployment and Operation Considerations
BRKSPG-2061	IPv6 Deployment Best Practices for the Cable Access Network
BRKCOL-2020	IPv6 in Enterprise Unified Communications Networks
BRKSEC-3003	Advanced IPv6 Security in the LAN
BRKRST-3123	Segment Routing for IPv6 Networks
BRKSEC-3200	Advanced IPv6 Security Threats and Mitigation
BRKRST-2301	Intermediate - Enterprise IPv6 Deployment

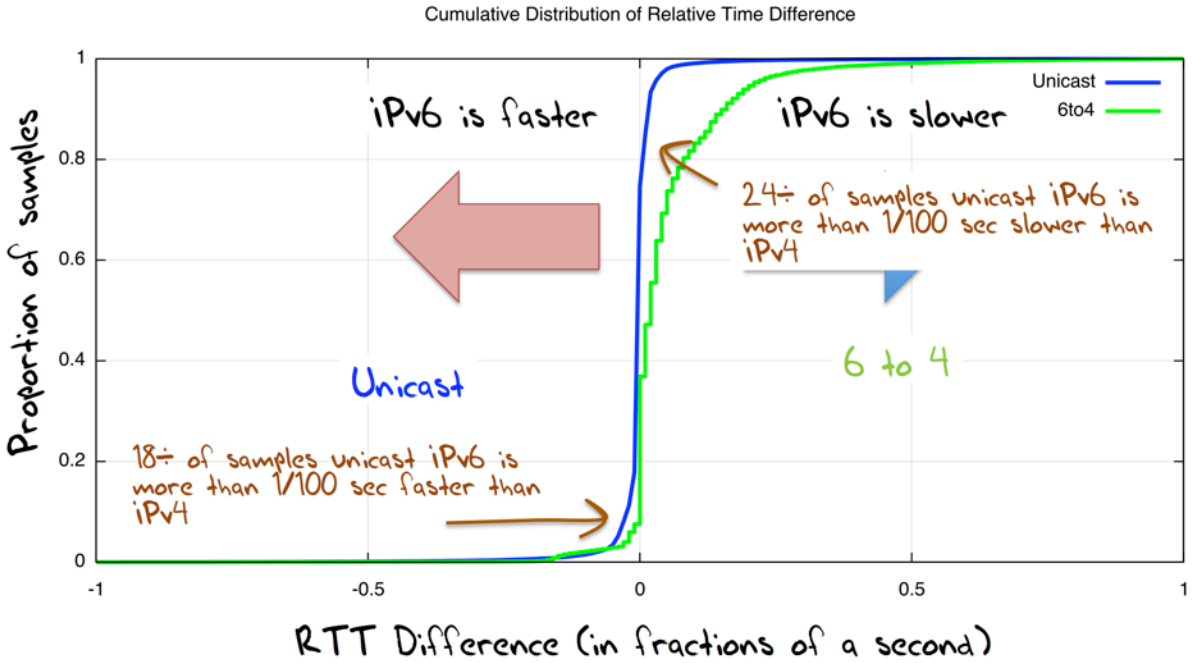
What is “IPv6-centric” ?

- BRKRST-2616 (Mark Townsley)
  - Use cases for IPv6 centric solutions in the network
  - Addressing Networking challenges with latest Innovations in IPv6
- "IPv6-first" design
- IPv4 compatibility: as a service
- IPv6-unique features

Why IPv6-centric ?

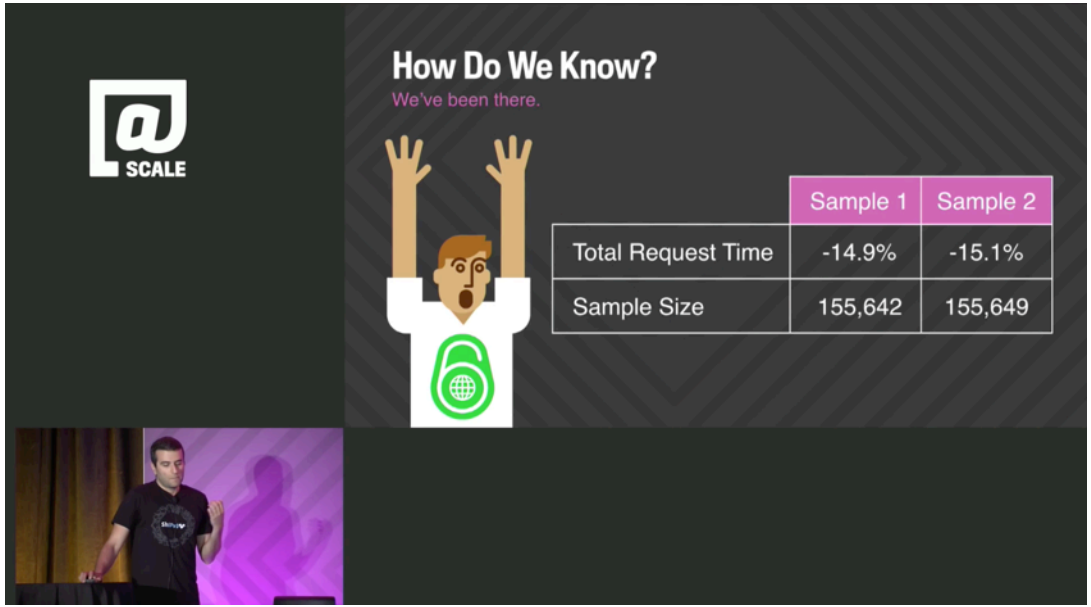
- Design app against one IP version easier
  - Dualstack = 2x the work
  - Still must support both of course – but understand priorities
    - Explore/focus on improvements/simplifications by IPv6
    - Support complexities of IPv4 as much as necessary
- Better IPv6 performance vs. IPv4
- Automation
- IPv4 sunset happening

Overall, IPv6 is as fast as IPv4 (except 6to4!)



https://ripe71.ripe.net/archives/video/1219

Except On Mobile: IPv6 is **15% faster** than IPv4



https://www.youtube.com/watch?v=_7rcAIbvzVY

Additionally: IPv6 Gets 25ms Bias on iOS 9

https://www.ietf.org/mail-archive/web/v6ops/current/msg22455.html

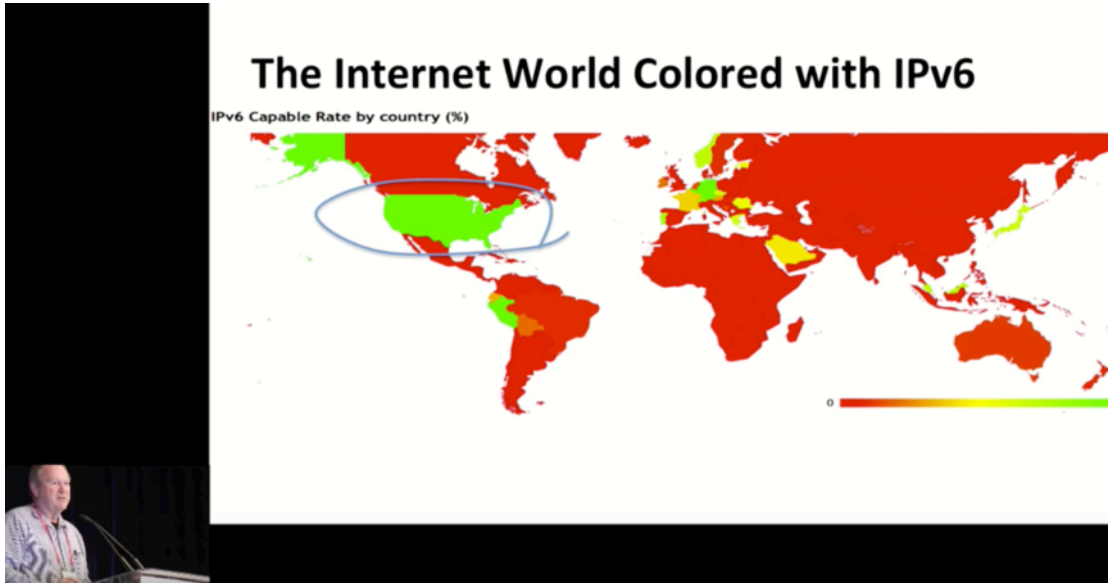
Because IPv6 support is so critical to ensuring your applications work across the world for every customer, we are making it an AppStore submission requirement, starting with iOS 9.”

Sebastien Marineau, VP of Core OS, Apple (June 2015)

“Starting June 1, 2016 all apps submitted to the App Store must support IPv6-only networking.”

https://developer.apple.com/news/?id=05042016a

Benefits of Deploying IPv6-Only: Operator Feedback



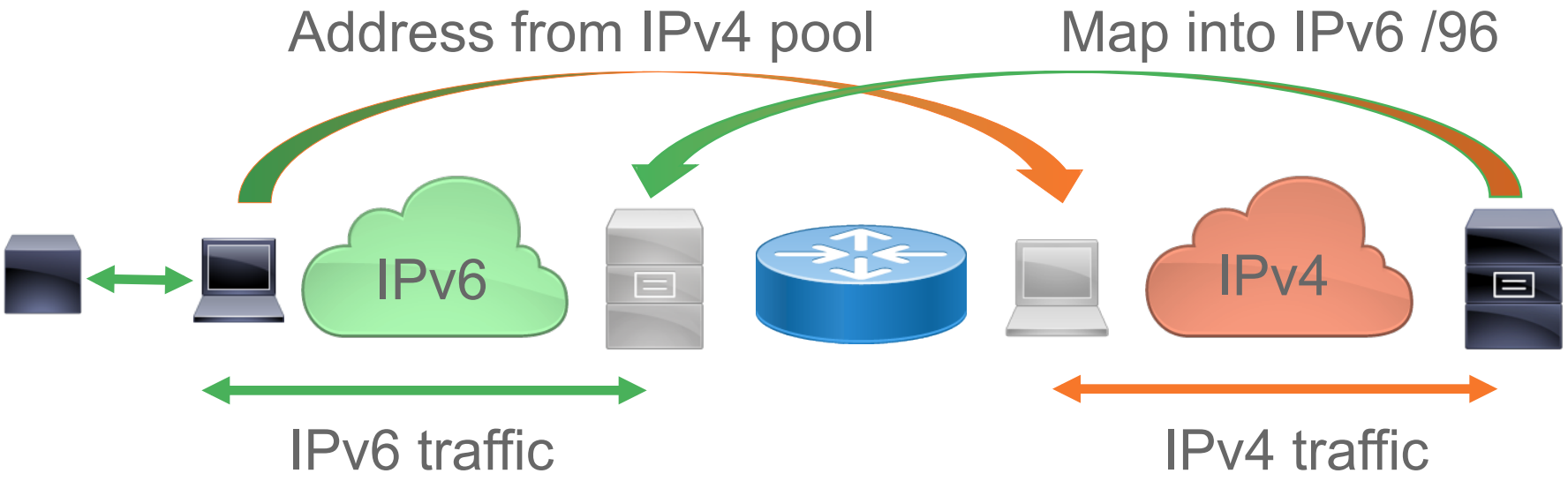
https://www.youtube.com/watch?v=EfjdOc41g0s

IPv6-only deployments: it's a reality

- IPV6-only clients
  - T-Mobile USA
    - http://www.internetsociety.org/deploy360/resources/case-study-t-mobile-us-goes-ipv6-only-using-464xlat/
  - Orange Poland
    - https://www.youtube.com/watch?v=Y0G5PTtZjTM (Polish language)
  - Telenor Norway (opt-in)
    - http://blog.toreanderson.no/2015/09/20/ipv6-mobile-roaming-possible-or-not.html
- IPv6-only servers
  - Redpill Linpro
    - http://blog.ipspace.net/2012/05/ipv6-only-data-center-built-by-tore.html

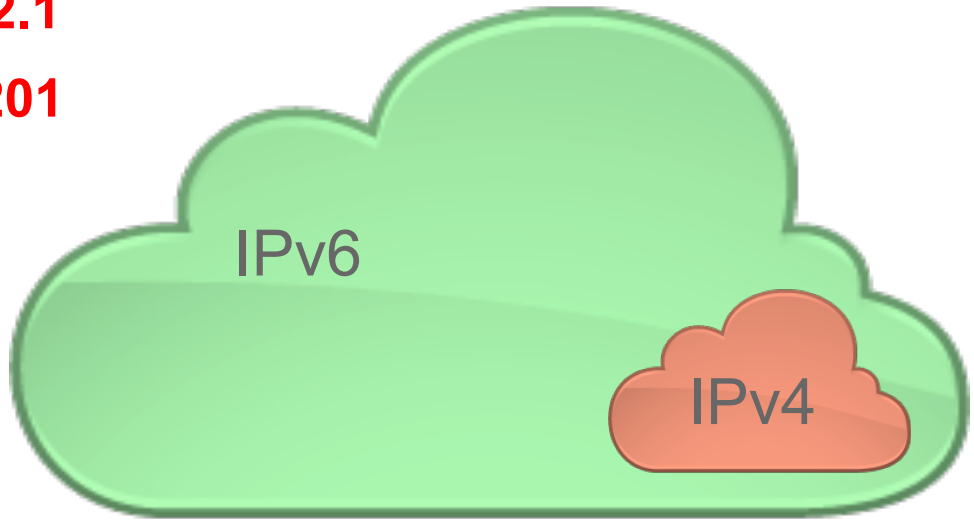
What is IPv6-centric: IPv6-only clients

NAT64 for an IPv6-only client

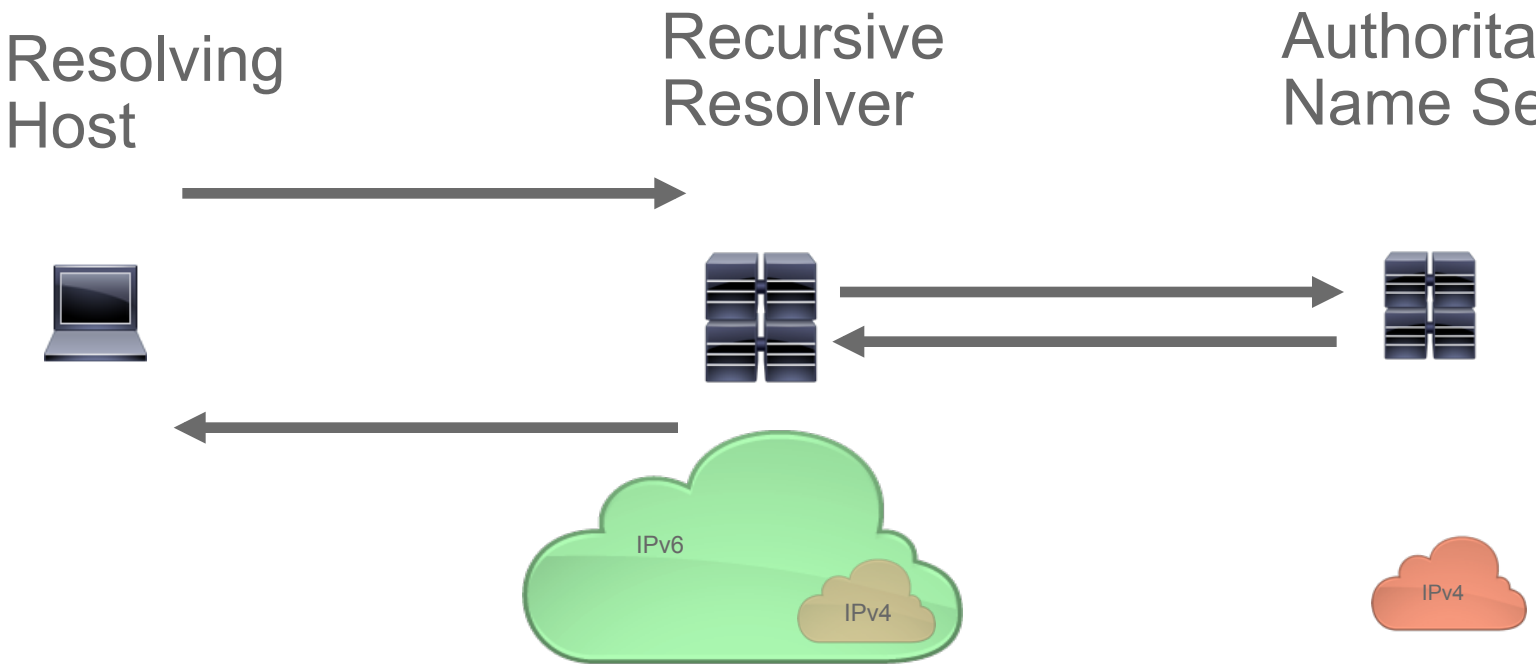


IPv4-embedded syntax for IPv6

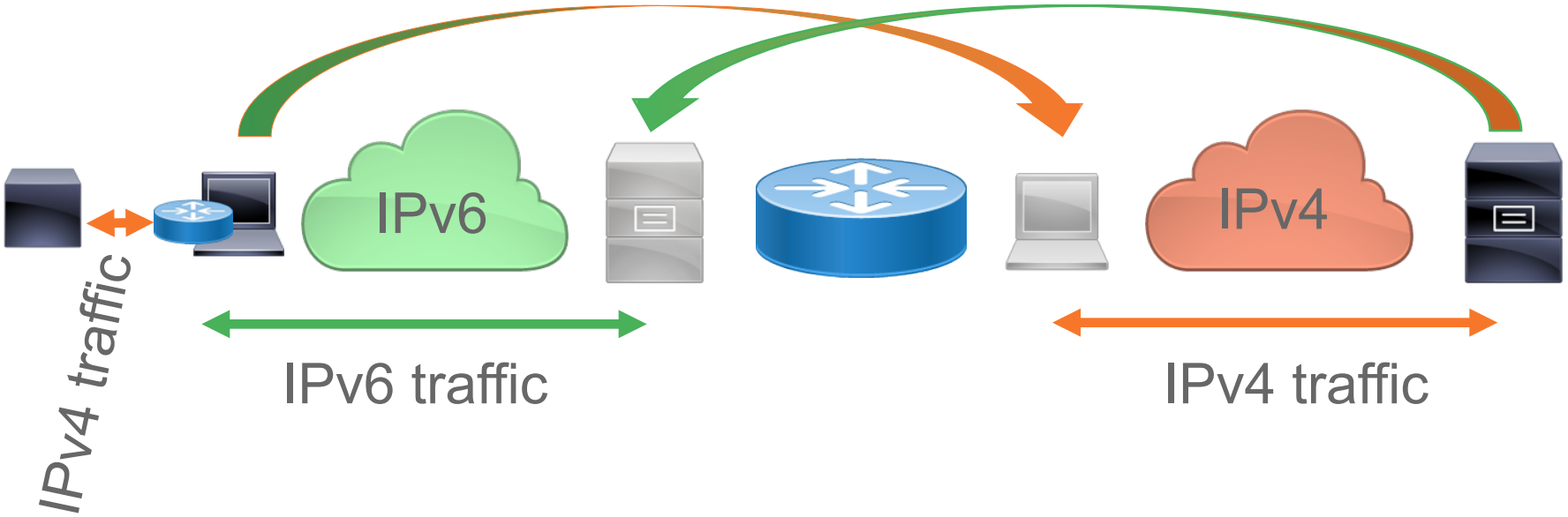
- 2001:db8:aaaa:aaaa::**192.0.2.1**
- 2001:db8:aaaa:aaaa::**c000:201**



DNS64 – Synthesize the addresses

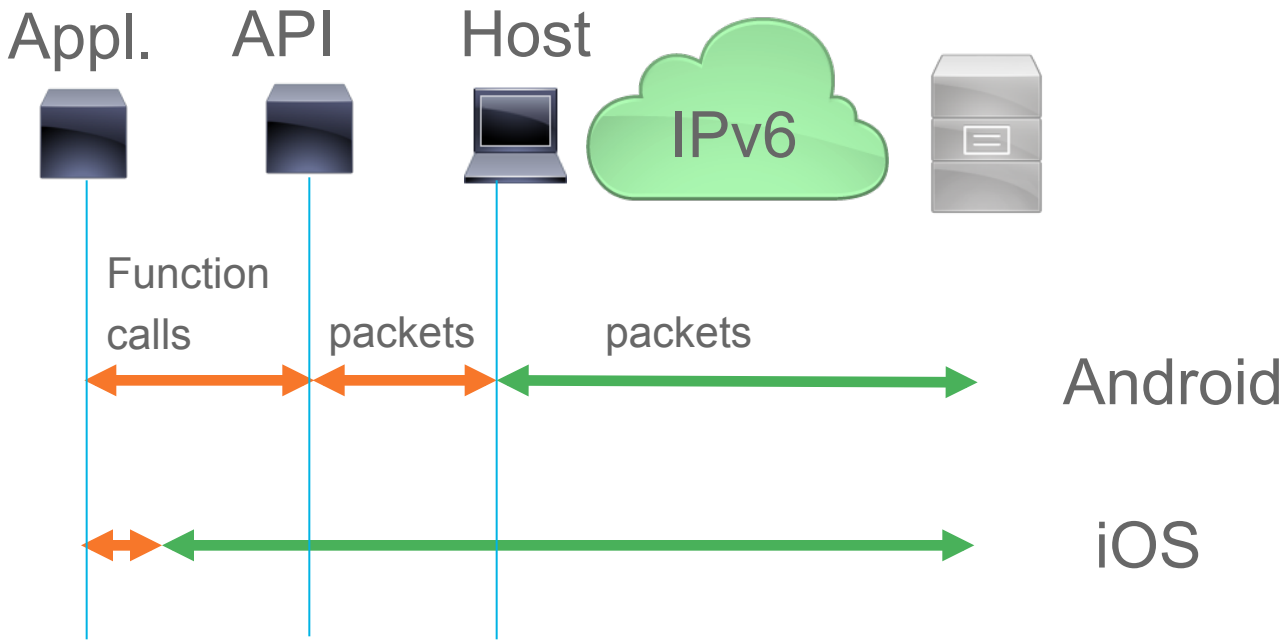


464XLAT: legacy apps "just work"



Do we need per-packet translation on client ?

iOS vs. Android approach for IPv4 communications

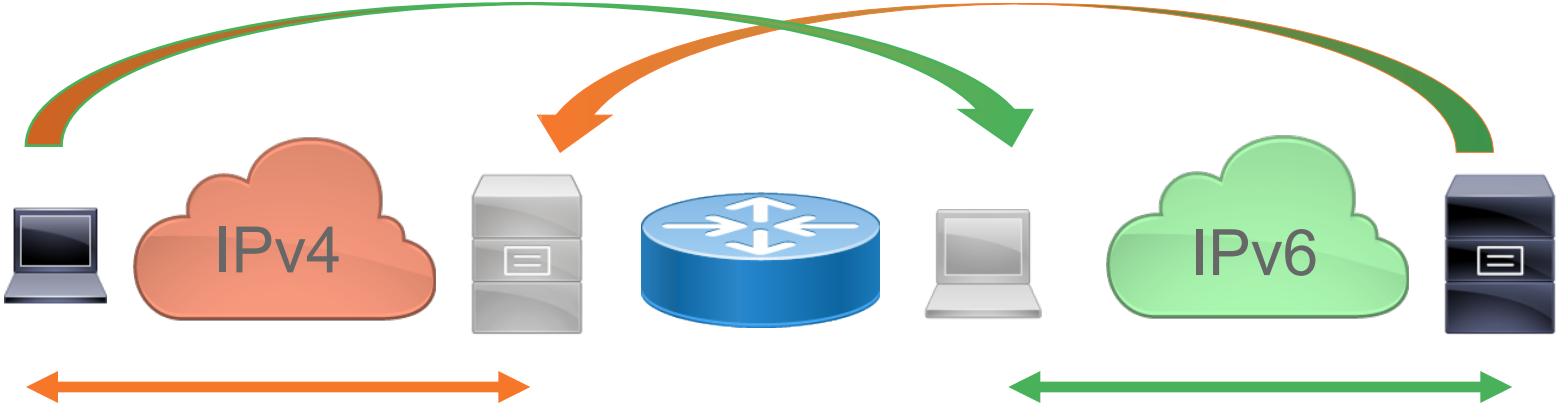


What is IPv6-centric: IPv6-only servers

IPv6-centric on the server

- Address management complexity
  - 2 stacks = easy to make mistake
- IPv4 exhaustion
  - Subnetting done in powers of 2 will waste usable addresses

SIIT-DC On The Server Side



https://tools.ietf.org/html/draft-ietf-v6ops-siit-dc-03

Stateful And Stateless NAT64: When To Use ?

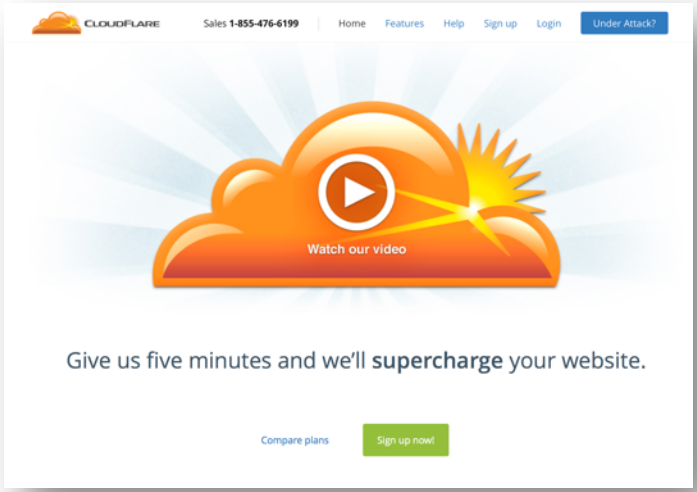
IPv6 clients

SIIT-DC

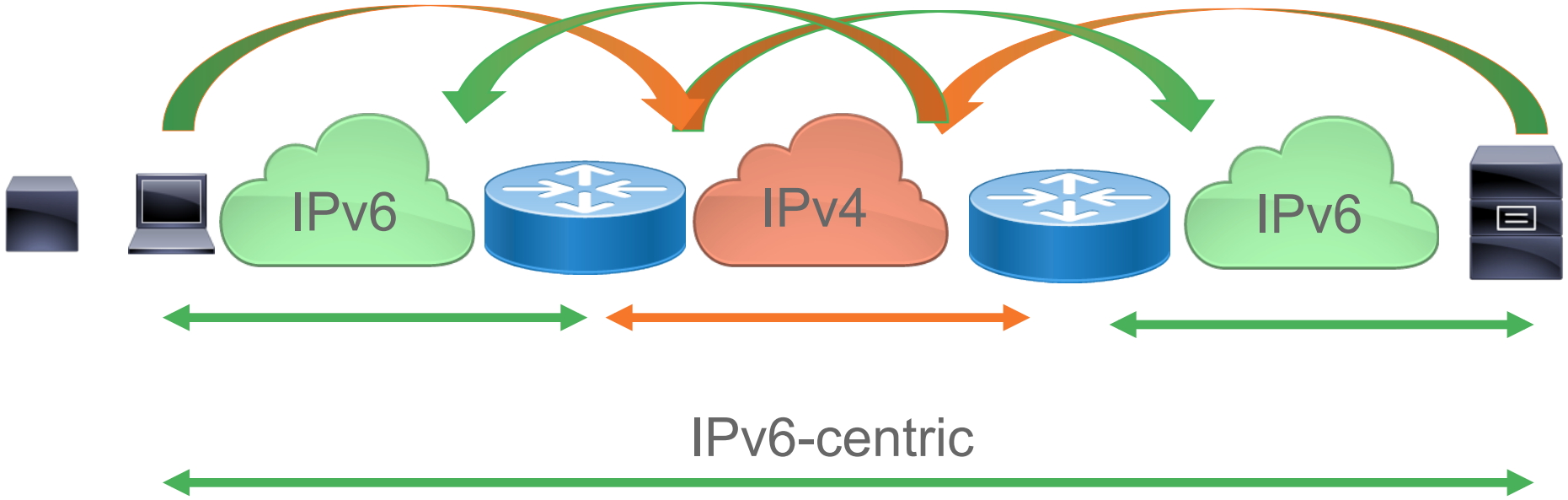
http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/enterprise-ipv6-solution/white_paper_c11-676278.html

IPv6-only: 2 Euro/year Personal Server

- That is server + IPv6 address
  - IPv4 address is 1 Euro/month (excluding server!)
- IPv6-only Lowendspirit VPS w/128mb
- Cloudflare free tier for IPv4 frontend + SSL



IPv6-centric: squeeze IPv4 out, transition



Languages and OSes

C and POSIX

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>

int
getaddrinfo(const char *hostname, const char *servname,
        const struct addrinfo *hints, struct addrinfo **res);

void
freeaddrinfo(struct addrinfo *ai);
```

The **getaddrinfo**() function is defined by the IEEE Std 1003.1-2004 (``POSIX.1'') specification and documented in RFC 3493, ``Basic Socket Interface Extensions for IPv6''.

Hints For Name Resolution

```
struct addrinfo {
            int ai_flags;              /* input flags */
            int ai_family;             /* protocol family for socket */
            int ai_socktype;           /* socket type */
            int ai_protocol;           /* protocol for socket */
            socklen_t ai_addrlen;      /* length of socket-address */
            struct sockaddr *ai_addr; /* socket-address for socket */
            char *ai_canonname;        /* canonical name for service location */
            struct addrinfo *ai_next; /* pointer to next in list */
    };
```

Some Interesting Values of ai_flags

- AI_ADDRCONFIG
  - Only return IPv4 addresses if IPv4 is present on interface
- AI_NUMERICHOST
  - The argument is a numeric address, do not attempt DNS resolution
- AI_PASSIVE
  - Allow for a listening socket: IN*ADDR_ANY if hostname is NULL
- AI_V4MAPPED
  - Return IPv4 addresses as IPv4-mapped IPv6

Tight Coupling of Addrinfo With Socket Open

```
getaddrinfo("www.kame.net", "http", &hints, &res0);
for (res = res0; res; res = res->ai_next) {
    s = socket(res->ai_family,res->ai_socktype,res->ai_protocol);
    if (s < 0) { cause = "socket"; continue; }

    if (connect(s, res->ai_addr, res->ai_addrlen) < 0) {
         cause = "connect"; close(s); s = -1;
         continue;
    }

    break;  /* okay we got one */
}
```

"lookup" and "connect" separate => problem

Sequential Connect

```
hints.ai_family = PF_UNSPEC;

getaddrinfo("www.kame.net",
"http", &hints, &res0);

for(res=res0;res;res=res->ai_next)
{
  s = socket(res->ai_family,
             res->ai_socktype,
              res->ai_protocol);

  connect(s, res->ai_addr,
             res->ai_addrlen)
```

```
struct addrinfo hints, *res, *res0;
int error;
int s;
const char *cause = NULL;

memset(&hints, 0, sizeof(hints));
hints.ai_family = PF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;
error = getaddrinfo("www.kame.net", "http", &hints, &res0);
if (error) {
        errx(1, "%s", gai_strerror(error));
        /*NOTREACHED*/
}
s = -1;
for (res = res0; res; res = res->ai_next) {
        s = socket(res->ai_family, res->ai_socktype,
            res->ai_protocol);
        if (s < 0) {
                cause = "socket";
                continue;
        }

        if (connect(s, res->ai_addr, res->ai_addrlen) < 0) {
                cause = "connect";
                close(s);
                s = -1;
                continue;
        }

        break;  /* okay we got one */
}
if (s < 0) {
        err(1, "%s", cause);
        /*NOTREACHED*/
}
freeaddrinfo(res0);
```

Listen On All AFs

```
hints.ai_family = PF_UNSPEC;
getaddrinfo(NULL, "http",
            &hints, &res0);

 for(res=res0;res &&
       nsock < MAXSOCK;
       res = res->ai_next) {
   s[nsock] = socket(
        res->ai_family,
        ….. );
   bind(s[nsock],
        res->ai_addr,
        res->ai_addrlen
```

```
struct addrinfo hints, *res, *res0;
int error;
int s[MAXSOCK];
int nsock;
const char *cause = NULL;

memset(&hints, 0, sizeof(hints));
hints.ai_family = PF_UNSPEC;
hints.ai_socktype = SOCK_STREAM;
hints.ai_flags = AI_PASSIVE;
error = getaddrinfo(NULL, "http", &hints, &res0);
if (error) {
        errx(1, "%s", gai_strerror(error));
        /*NOTREACHED*/
}
nsock = 0;
for (res = res0; res && nsock < MAXSOCK; res = res->ai_next) {
        s[nsock] = socket(res->ai_family, res->ai_socktype,
            res->ai_protocol);
        if (s[nsock] < 0) {
                cause = "socket";
                continue;
        }
        if (bind(s[nsock], res->ai_addr, res->ai_addrlen) < 0) {
                cause = "bind";
                close(s[nsock]);
                continue;
        }
        (void) listen(s[nsock], 5);
        nsock++;
}
if (nsock == 0) {
        err(1, "%s", cause);
        /*NOTREACHED*/
}
freeaddrinfo(res0);
```

Questions Unanswered With Basic API

- Near-simultaneous open ? (RFC6555)
  - Need your own higher-layer library
- Source address selection (if different prefixes)
  - Might need to bind sockets explicitly

Python

- Standard socket interface similar to C
- Async frameworks
  - Asyncore
  - Twisted

Python Server program

for res **in** socket.getaddrinfo(HOST, PORT, socket.AF_UNSPEC, …

try:
    s = socket.socket(af, socktype, proto)
    s.bind(sa)

```
# Echo server program
import socket
import sys

HOST = None               # Symbolic name meaning all available interfaces
PORT = 50007              # Arbitrary non-privileged port
s = None
for res in socket.getaddrinfo(HOST, PORT, socket.AF_UNSPEC,
                              socket.SOCK_STREAM, 0, socket.AI_PASSIVE):
    af, socktype, proto, canonname, sa = res
    try:
        s = socket.socket(af, socktype, proto)
    except socket.error as msg:
        s = None
        continue
    try:
        s.bind(sa)
        s.listen(1)
    except socket.error as msg:
        s.close()
        s = None
        continue
    break
if s is None:
    print 'could not open socket'
    sys.exit(1)
conn, addr = s.accept()
print 'Connected by', addr
while 1:
    data = conn.recv(1024)
    if not data: break
    conn.send(data)
conn.close()
```

https://docs.python.org/2/library/socket.html

Python Client program

for res **in** socket.getaddrinfo(HOST, PORT, socket.AF_UNSPEC, …

```
# Echo client program
import socket
import sys

HOST = 'daring.cwi.nl'    # The remote host
PORT = 50007              # The same port as used by the server
s = None
for res in socket.getaddrinfo(HOST, PORT, socket.AF_UNSPEC, socket.SOCK_STREAM):
    af, socktype, proto, canonname, sa = res
    try:
        s = socket.socket(af, socktype, proto)
    except socket.error as msg:
        s = None
        continue
    try:
        s.connect(sa)
    except socket.error as msg:
        s.close()
        s = None
        continue
    break
if s is None:
    print 'could not open socket'
    sys.exit(1)
s.sendall('Hello, world')
data = s.recv(1024)
s.close()
print 'Received', repr(data)
```

Asyncore

self.create_socket(socket.AF_INET, socket.SOCK_STREAM)

```
import asyncore, socket

class HTTPClient(asyncore.dispatcher):

    def __init__(self, host, path):
        asyncore.dispatcher.__init__(self)
        self.create_socket(socket.AF_INET, socket.SOCK_STREAM)
        self.connect( (host, 80) )
        self.buffer = 'GET %s HTTP/1.0\r\n\r\n' % path

    def handle_connect(self):
        pass

    def handle_close(self):
        self.close()

    def handle_read(self):
        print self.recv(8192)

    def writable(self):
        return (len(self.buffer) > 0)

    def handle_write(self):
        sent = self.send(self.buffer)
        self.buffer = self.buffer[sent:]


client = HTTPClient('www.python.org', '/')
asyncore.loop()
```

https://docs.python.org/2/library/asyncore.html

Twisted: http://twistedmatrix.com/trac/ticket/3014

Image source: http://www.amazon.com/Its-Complicated-Meryl-Streep/dp/B0038N9WKU

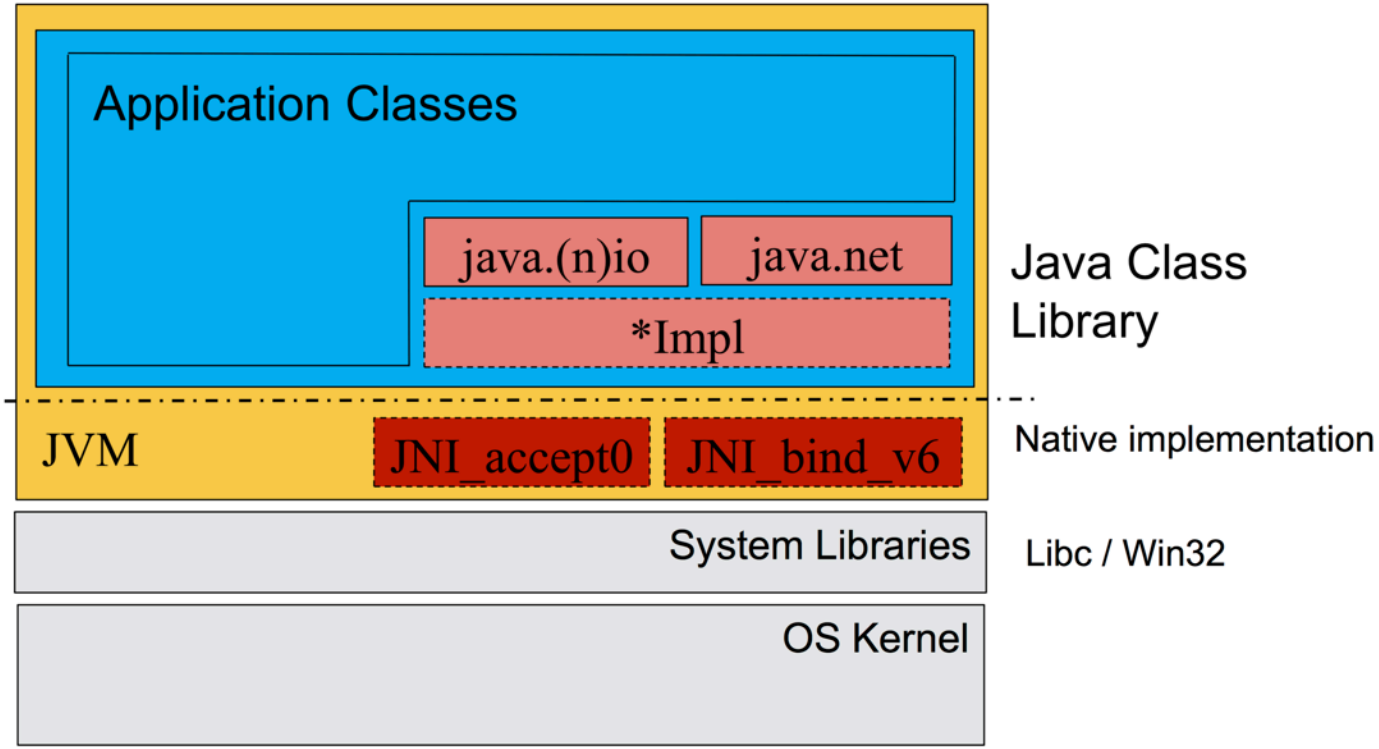
Python3.3: ipaddress

"The functions and classes in this module make it straightforward to handle various tasks related to IP addresses…"

https://docs.python.org/3/library/ipaddress.html

```
>>> ipaddress.ip_address('192.168.0.1')
IPv4Address('192.168.0.1')
>>> ipaddress.ip_address('2001:db8::')
IPv6Address('2001:db8::')
```

Java



http://bernd.eckenfels.net/files/IPv6Con%20Java%20Apps%20IPv6%20fit%20machen.pdf

Java

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

http://bernd.eckenfels.net/files/IPv6Con%20Java%20Apps%20IPv6%20fit%20machen.pdf

PHP

- Mostly run in high-level context
- Using the hostnames should normally do the job
- Biggest care: address literals
  - Avoid if you can!

PHP: built-in library example

bool socket_connect (resource $socket ,
string $address [, int $port = 0])

The **address** parameter is either an IPv4 address in dotted-quad notation (e.g. *127.0.0.1*) if **socket** is **AF_INET**, a valid IPv6 address (e.g. *::1*) if IPv6 support is enabled and **socket** is **AF_INET6**

http://php.net/manual/en/function.socket-connect.php

PHP: how NOT to validate addresses

```
function isIPv6($ip) {
  if(   strpos($ip, ":") !== false &&
        strpos($ip, ".") === false) {
     return true; //Pure format
  }
  elseif(strpos($ip, ":") !== false &&
         strpos($ip, ".") !== false){
    return true; //dual format
  }
  else{
    return false;
  }
}
```

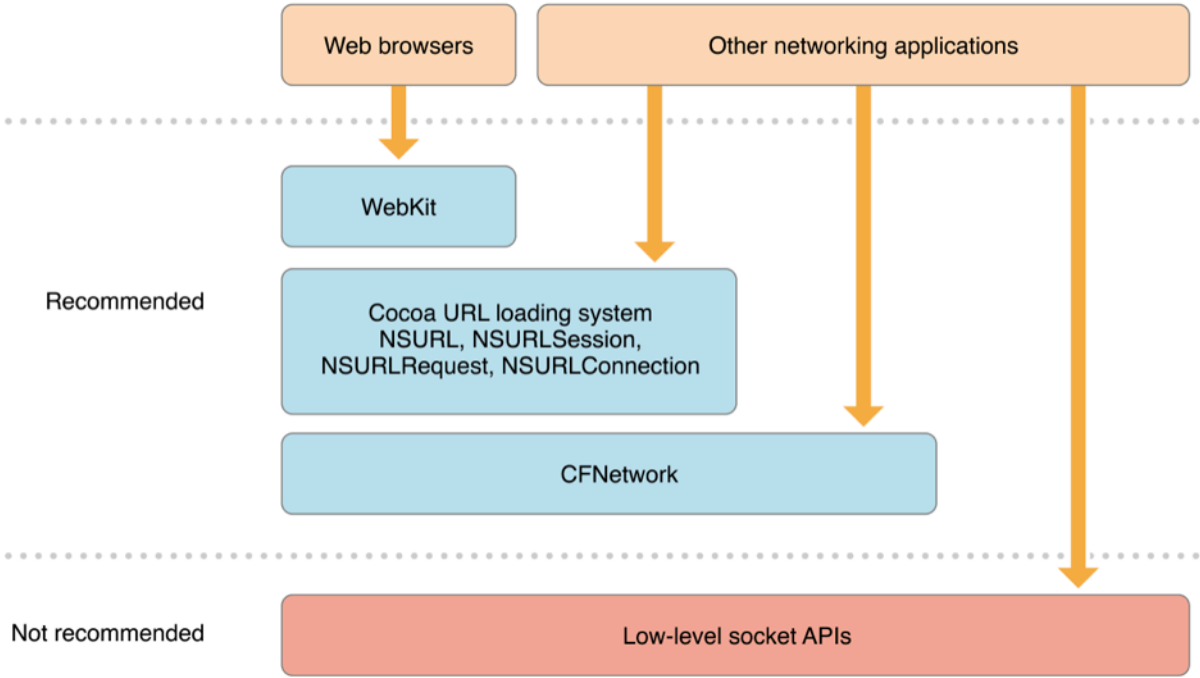
Is 12:45 an address ?

PHP: an example of using filter_var()

```
function isIPv6($ip) {
   if(filter_var($ip, FILTER_VALIDATE_IP)) {
       if(filter_var($ip, FILTER_VALIDATE_IP, FILTER_FLAG_IPV6)) {
          //It is IPv6 indeed.
        } else {
          //It is IPv4
       }
   } else {
      // NOT VALID IP
   }
}
```

https://github.com/php/php-src/blob/master/ext/filter/logical_filters.c#L746

iOS Networking Frameworks summary



High vs. Low level APIs

- Socket APIs considered “low level”
  - Mostly standardized – but lot of small platform differences
  - Previous slide examples from programming languages all “Socket API”
- Vendors have higher level APIs
  - Simplify development even further
  - But run the risk of making code even harder to port
- Facebook client library code
  - Cross platform higher level API set

iOS / OSX

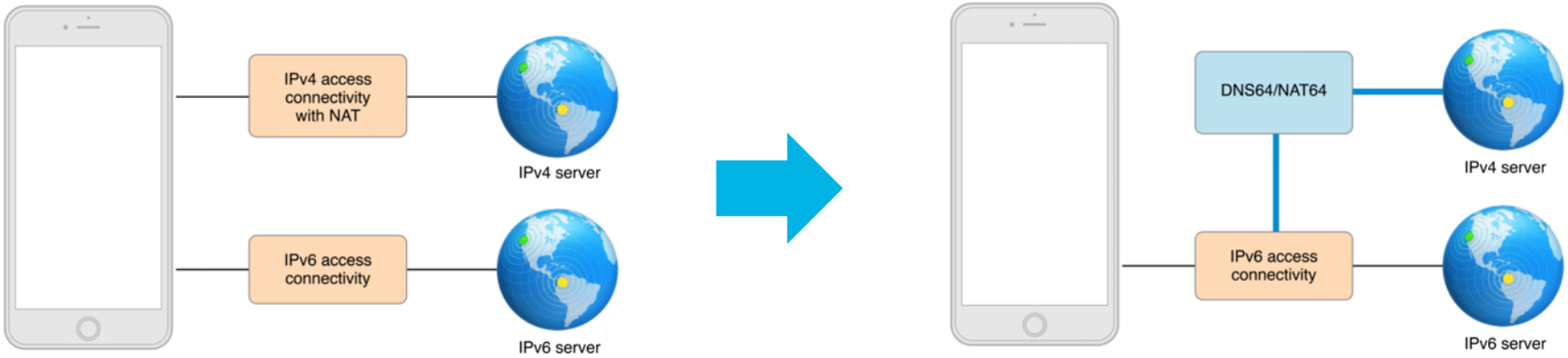
The preferred way to connect to a host is with an API that accepts a DNS name, such as CFHost or CFNetService.

https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/NetworkingOverview/CommonPitfalls/CommonPitfalls.html

iOS: A Typical Way To Open Network Connection

```
- (void)initNetworkCommunication {
    CFReadStreamRef readStream;
    CFWriteStreamRef writeStream;
    CFStreamCreatePairWithSocketToHost(NULL,
        (CFStringRef)@"localhost", 80,
        &readStream, &writeStream);
    inputStream = (NSInputStream *)readStream;
    outputStream = (NSOutputStream *)writeStream;
}
```

iOS: Supporting NAT64+DNS64



https://developer.apple.com/library/prerelease/ios/documentation/NetworkingInternetWeb/Conceptual/NetworkingOverview/UnderstandingandPreparingfortheIPv6Transition/UnderstandingandPreparingfortheIPv6Transition.html

iOS IPv4-only functions

inet_addr()
inet_aton()
inet_lnaof()
inet_makeaddr()
inet_netof()
inet_network()
inet_ntoa()
inet_ntoa_r()
bindresvport()
getipv4sourcefilter()
setipv4sourcefilter()

User Interface
Address, Names and URLs

Names

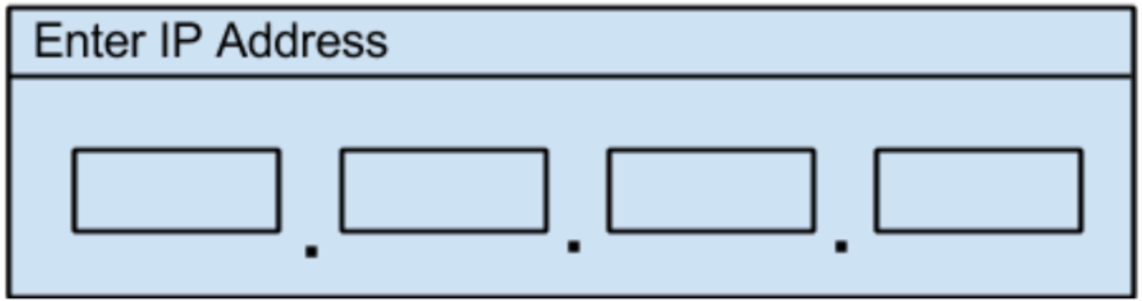
- DO NOT LET USERS DEAL WITH IPv6 ADDRESSES
  IPv4 addresses where barely acceptable.
- If you do need more than static-DNS defined names for IPv6 addresses (AAAA):
  - Use DNS-SD: “DNS Service Discovery”
    - Dynamic name binding (AAAA) for addresses
    - Names for (addrs,port) – called service names
    - Includes also mDNS – DNS via link-local multicast (ad-hoc)
  - Bonjour and Avahi SDKs – both multi-platform, open source
- Name pros:
  - Same in IPv4 and IPv6, Easy to remember, Works with NAT64+DNS64
- Cons
  - 3rd party signaling across split-DNS boundary (how many apps have 3rd party signaling?)

Names (2)

- DO NOT LET DEVELOPERS DEAL WITH IPv6 ADDRESSES
  - See last section - SDK issue
- DO NOT LET NETWORK OPERATORS DEAL WITH IPv6 ADDRESSES
  - SDN controllers to the rescue ?!

Input fields and addresses

- Addresses can be short and long
- Canonical format, lower case / uppercase
- Link-local addresses



IPv6 and URLs - Ambiguity #1: ":"

- https://www.ietf.org/rfc/rfc2732.txt

```
http://[2001:0:0:0:8:800:200C:417A]/index.html
http://[2001:2a00:100:7031::1]
http://[2001::8:800:200C:417A]/foo
http://[::192.9.5.5]/ipng
http://[::FFFF:129.144.52.38]:80/index.html
http://[2010:836B:4179::836B:4179]
```

Link-Local addresses and URLs - Ambiguity #2: "%"

- % is used for %-encoding in URIs: ambiguity
- "Be liberal with what you accept" principle

"fe80::a%en1"

"fe80::a%25ee1"

https://bugzilla.mozilla.org/show_bug.cgi?id=700999

https://tools.ietf.org/html/rfc6874

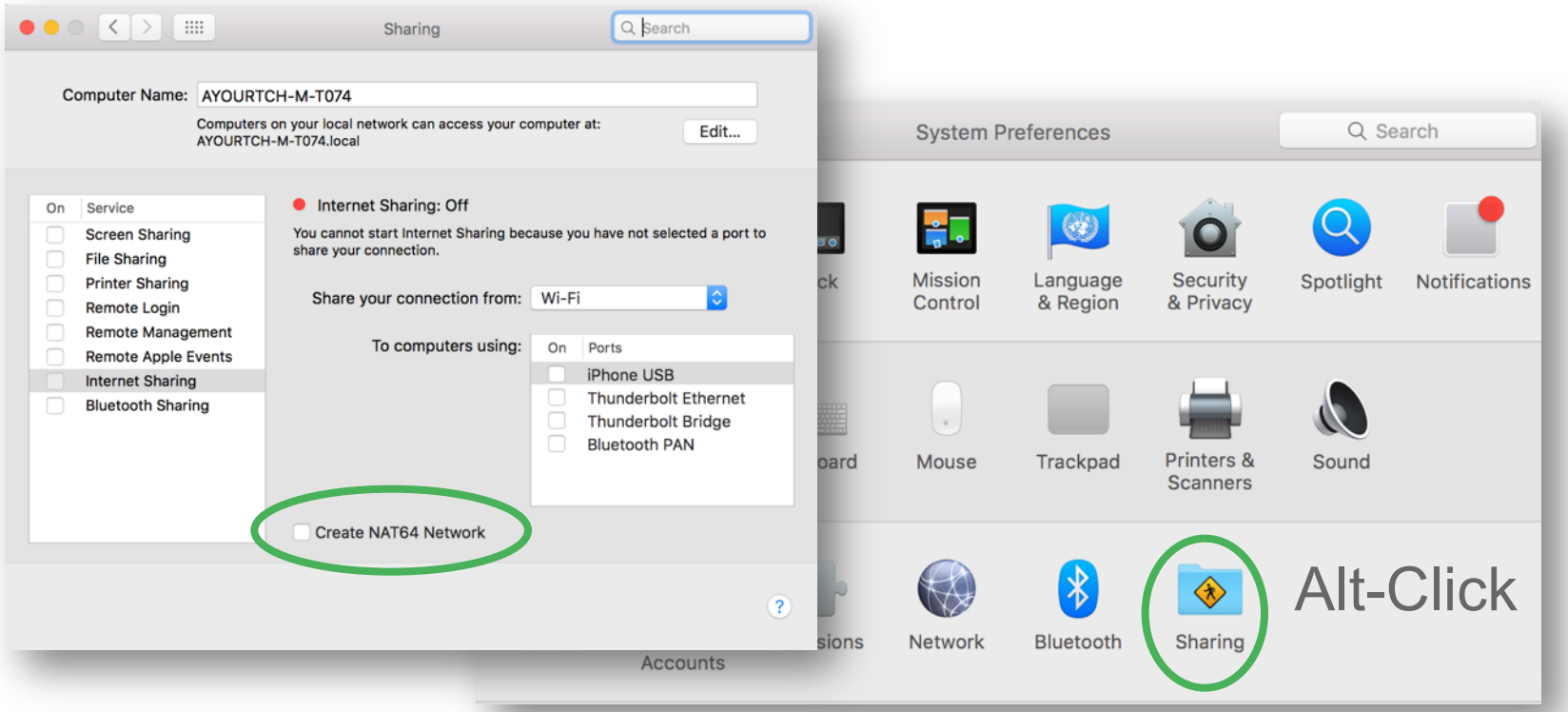
Testing Your Applications

Testing NAT64 client applications

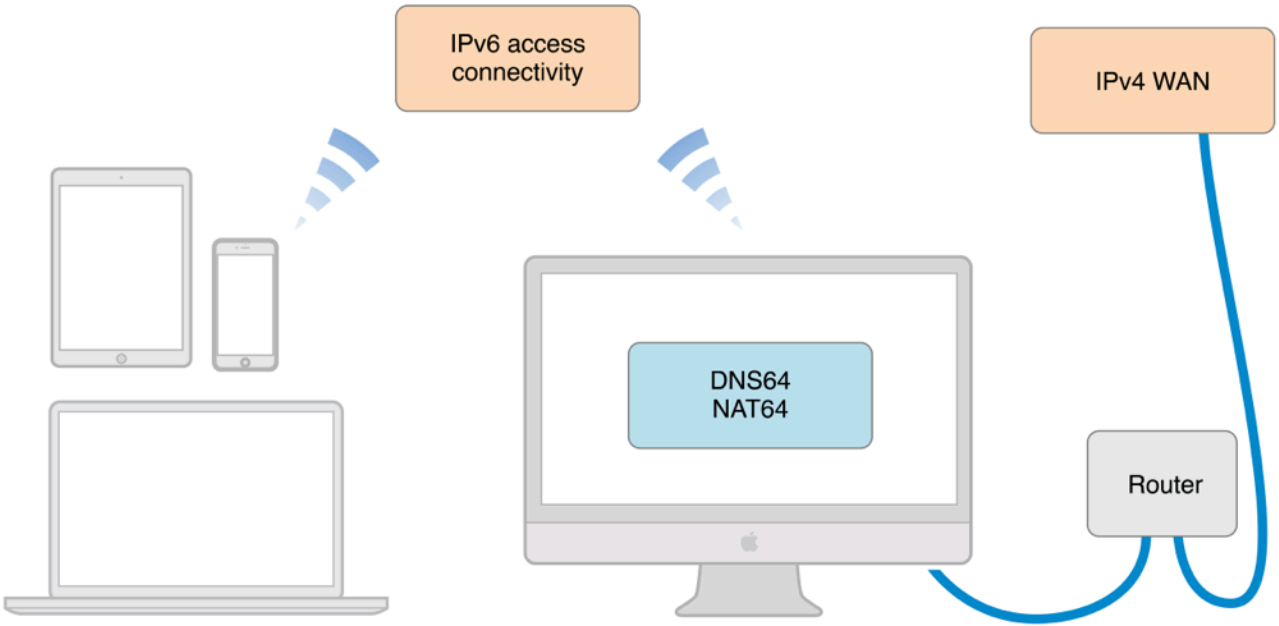
```
ipv6 access-list NAT64
  permit tcp 2001:DB8::/64 64:FF9B::/64
  permit udp 2001:DB8::/64 64:FF9B::/64
  permit icmp 2001:DB8::/64 64:FF9B::/64
 !
 !
 nat64 v4 pool NAT64-IPv4 192.0.2.1 192.0.2.1
 nat64 v6v4 list NAT64 pool NAT64-IPv4 overload
 !
```

http://docwiki.cisco.com/wiki/IPv6_only_setup_with_NAT64

Have A Mac (with 10.11) ? Have IPv6-Only Network!

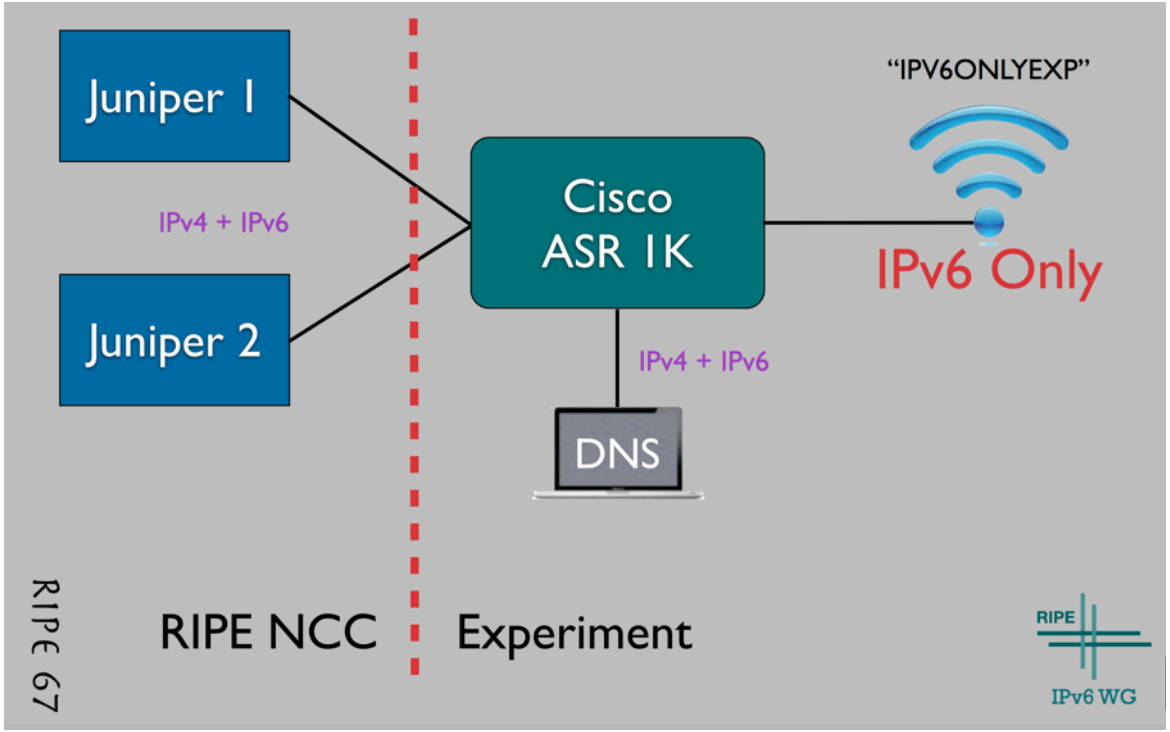


OS X El Capitan (10.11) as access gateway



Communities as a "real-world lab"

RIPE IPv6-only/NAT64 network



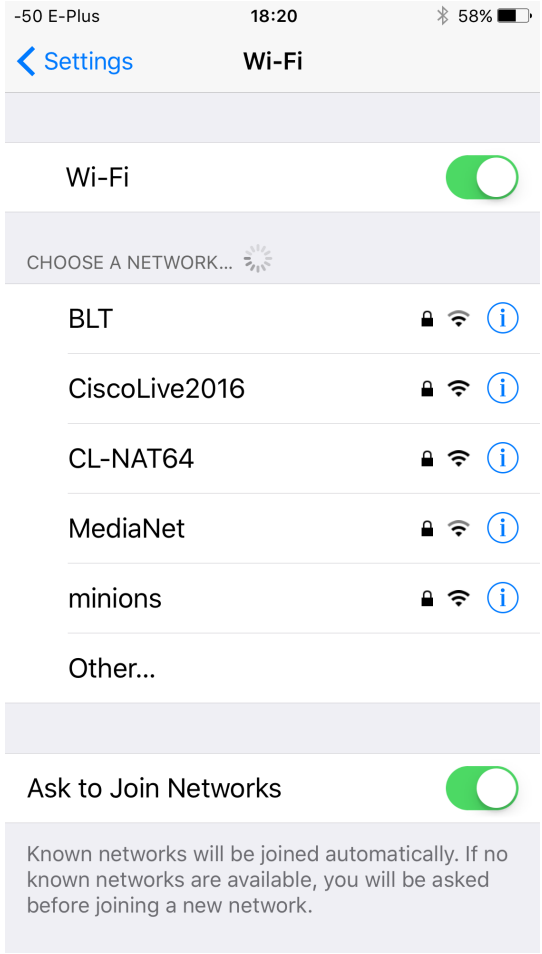
Lots of active feedback

Cisco

- supportforums.cisco.com
  - Operator focused
  - IPv6 integration and transition
    - https://supportforums.cisco.com/community/5531/ipv6-integration-and-transition
  - Also: IPv6 covered as part of different communities (IPv6 in technology FOOBAR)
- Devnet Communities:
  - Application developer focused.
  - https://communities.cisco.com/community/developer/ipv6
    - Just created. Will start populating with content. Please join!
- How else can Cisco help you develop ipv6 centric apps ?

CiscoLive IPv6-only network

- Why ?
  - Test how app/net will work when the sun has set on IPv4!
- SSID: “CL-NAT64”
- WPA2-PSK
  - Key: “cl-nat64”
- Stateless DHCPv6 + RDNSS
- Feedback/questions:
  - Twitter #CLNAT64
  - Or direct @ayourtch ☺



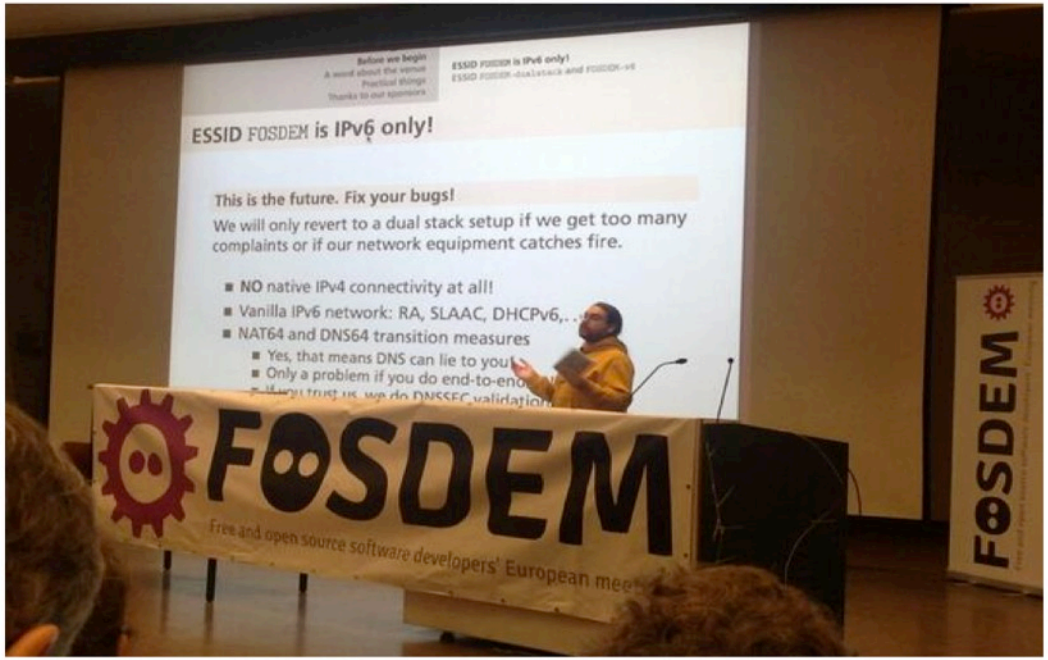
(Probably) The World First

IPv6-only/NAT64 WiFi **by default**

Dirk Haun @dirkhaun 6h

Forced **progress**: #**FOSDEM** network is IPv6 only by default. "We're all developers. If you find bugs, fix them!"
pic.twitter.com/pM9vvVYB9e

Hide photo Reply Retweet Favorite More



Technical details

- FOSDEM SSID (default): IPv6-only + NAT64
- FOSDEM_legacy SSID : dualstack

- ~15 people (volunteers) in the NOC
- Communication, Communication, Communication
  - Critical part of ensuring the expectations are correctly set

Maksim Melnikau @max_posedon 1h

So, I'm switched to **ipv6** **#fosdem** ESSID ... It is my first time ever when I use **ipv6** ... pic.twitter.com/KO1nCoCwHr

Expand Reply Retweet Favorite More

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::863a:4bff:fece:9dec  prefixlen 64  scopeid 0x20<link>
        inet6 2001:67c:1810:f051:863a:4bff:fece:9dec  prefixlen 64  scopeid 0x0<global>
        ether 84:3a:4b:ce:9d:ec  txqueuelen 1000  (Ethernet)
        RX packets 282214  bytes 262806444 (250.6 MiB)
        RX errors 0  dropped 6  overruns 0  frame 0
        TX packets 250257  bytes 46739319 (44.5 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Olaf Flebbe @OlafFlebbe 2h

#fosdem vmware fusion seems not to work with **ipv6**

t Favorite More

Łukasz Jernaś @didzej1 16 Feb 2011

I'm all for being **#IPv6** only at next years **#FOSDEM**, this time I'll take a device with WiFi :P http://ur1.ca/39f5g

Expand Reply Retweet Favorite More

Grégory Paul @paulgreg 5h

At **#FOSDEM** with @thierrymarianne. Network is **IPv6** only and pretty solid for now ! Yeah #wifi #conference #rocks

Expand Reply Retweet Favorite More

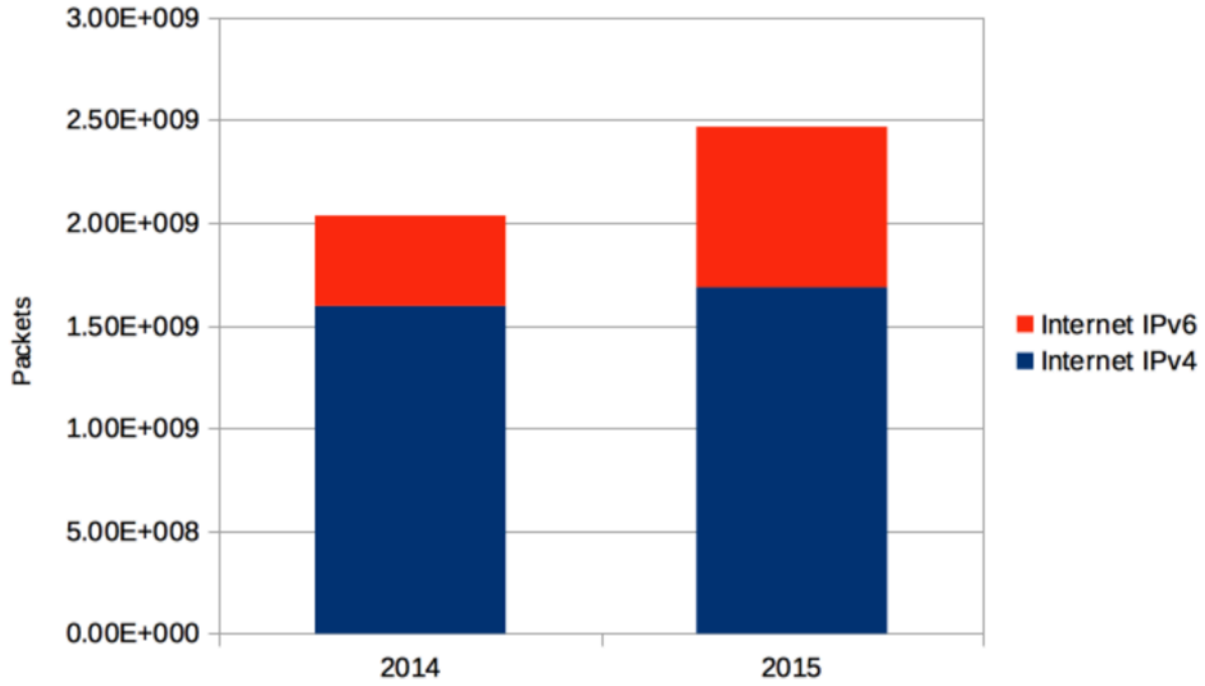
Andreas Olsso

Just cancelled n

IPv6 support. Instead using git-annex assistant - git-annex.branchable.com/assistant/

Expand Reply Retweet Favorite More

Internet-bound traffic from FOSDEM conference



Percentage of the clients using default SSID

- 2014: ~ 16%
- 2015: ~ 35%
- 2016: ~ 50%

- NOTE: Client base is Android-heavy!
- NOTE: These are not “IPv6” developer.
  These are not even “Network” developers
  Just “Application” developers
- Reference: http://blogs.cisco.com/getyourbuildon/fosdem-2016-a-first-quick-look

IPv6-only: Not Just For Networking Geeks!

CEDEC

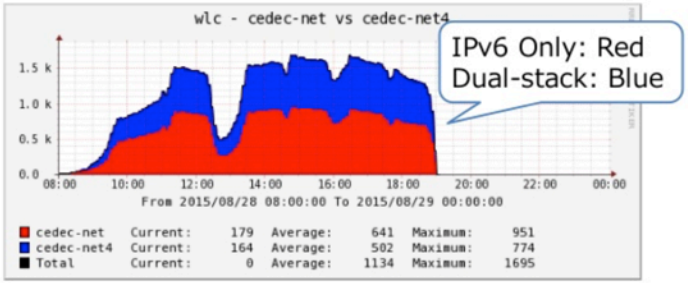
Clip slide

- Largest conference in Japan for computer entertainment developers
- 3 days at Pacifico Yokohama
- 6,373 attendees in 2015

Reach Next Level

Devices in each network

approx. **65% Devices**
works with **IPv6 Only**



almost same as share of Apple devices
If RDNSS is enabled, Android devices can be added

Reach Next Level

http://www.slideshare.net/yuyarin/janog37-ltcedecnet2015-en-57359924

Advanced considerations Fragmentation

IPv6 and Fragmentation

- IPv6 does not have "DF" bit
- Reliance on PMTUD
  - Blocked ICMPv6 is lethal.
  - Blocked fragments lethal too.
    - Uncommon to happen in IPv6
    - Rumors that this happens in the Internet sometimes for IPv6 (bad!).
- Min permitted link MTU = 1280
  - IPv4: 512.
  - Practically, all links today are >= 1500, so IPv6 Min-MTU packets + “any” tunnels fit.
  - Efficient enough to always only send <= 1280 to avoid PMTUD in IPv6
    - Author thinks: Yes. Also: Fragment overhead higher (IPv6 header vs. IP header)
  - IoT/low-energy/bitrate networks would like << 1280 MTU
    - IETF work ongoing

IPv6 and Fragmentation

- IP Multicast:
  - There is no PMTUD for IP multicast
    - Assume there was PMTUD for eg: 100,000 receiver in a tree. What is the result ?
    - At least one receiver will likely only have minimum MTU.
    - IPv4 experience: Every receiver had L1 path MTU 1500 + header (PPoE + VPN + …). IPTV video senders all set for MTU of <= 1400.
- Advanced IPv6 Socket API: RFC4542
  - Default unicast: Perform PMTUD
  - Multicast: Defaults to Max packet < 1280 (RFC4542)
- Recommendations
  - Avoid PMTUD unicast/multicast
  - UDP Unicast/Multicast: Never send packet > 1280
  - TCP: Set MSS to 1220 to avoid fragmentation.

https://labs.ripe.net/Members/gih/evaluating-ipv4-and-ipv6-packet-fragmentation

Advanced Considerations IPv6 addressing

Addresses, addresses and more addresses!

- IPv6 has 2^96 more addresses than IPv4
  - And 1000 new ways how to use them
  - To simplify & enhance applications and network deployment
  - To solve all the issues we did not even dare to tackle with IPv4
    - (or failed miserably)

http://streamd.hitparade.ch/cdimages/trini_lopez-if_i_had_a_hammer_(live)_s.jpg

Your personal, subscription dose of IPv6 addresses

Order your monthly refills now!

Lets Start Simple: IPv6 to simplify & improve IP Multicast apps

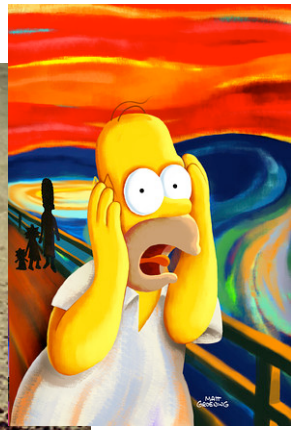
Challenge	Solution in IPv4	Solution in IPv6
Scoping: limit how "far" multicast can go – building, campus, enterprise,…	Define your scopes. Figure out your own IPv4 address ranges for each "scope". *Configure on all scope edge routers ACLs with scope address ranges*. Tell app-developers what address ranges are. Configure address-ranges into apps for 3rd party apps.	Scopes 2 (link-local) … 14 (Internet) defined in IPv6 architecture. Encoded as 4 bits in IPv6 multicast address. App just sets those bit in the addr. accordingly to what it wants. *Scope edge routers just configured with n=3..14*.
ASM (classic multicast – Any Source Multicast) addressing	Tight address space. Scoped address (like RFC1918) badly useable (collisions with other installations). Allocation of well-known (eg: hardcoded) group address for applications painful process with IANA. Officially not even permitted to use an IPv6 (global) multicast address for apps that are just running eg: within an enterprise.	Self-allocate global (and per-scope) unique IPv6 multicast addresses via RFC3306: Mechanism to construct IPv6 multicast group addresses (2^32 different ones – per scope) from any IPv6 unicast address range you own.
Source Specific Multicast (SSM)	Only global SSM addresses standardized. Invent your own address ranges for scoped SSM	SSM address range has same 4-scope bits, so address for all scopes available (RFC3306)
ASM / PIM-SM	Deploy protocols such as AutoRP or BSR on every router so routers learn the RP (Rendezvous Point). Or manually configure RP on every router.	Embedded-RP (RFC3956) defined multicast group addresses (derived from RFC3306) that also include the IPv6 unicast address of the RP – no protocols or per-router config needed.
Interdomain ASM / PIM-SM	No Internet standard. Must use MSDP which is just experimental (insecure, does not scale,…)	Just use Embedded-RP. No additional Interdomain work needed.

And Now For Something Completely Different

IPv6 unicast addressing

Source: http://www.bbc.co.uk/programmes/b00n7sf5

Source: Matt Groening



Link Local Addresses

- Ad-Hoc / Legacy “LAN” applications:
  - Built against ethernet, no IP: avoid manual address configuration.
  - Ad-Hoc networking more interesting now: IoT (eg: in home), Mesh networks,…
- IPv6 link-local addresses (LL)
  - Automatically assigned. All you need to use when just talking across “LAN”.
  - All hop-by-hop IPv6 protocols/solutions use LL.
  - Multiple interfaces: Ensure to include interface name in API calls
  - Ensure your app remembers interface for each neighbor (LL not unique across LANs).
- IPv4
  - LL defined as afterthought: RFC 3927 – 169.254.1.x
  - Not very useful: only initialized INSTEAD of routeable addresses

Unique Local Addresses (ULA) – RFC 4193

- IPv6 version of “private” addresses (IPv4: RFC1918)
  - Routeable / “global”
  - **Prohibited from being routed across Internet**
    - Must be filtered by ISPs
- Benefits (over IPv4):
  - No running out of private address space (if you use it right)
  - Low probability of “collision” for common IPv4 RFC1918 problem:
    - ULA is: 40 bit random prefix, 16 bit “subnet”, 64 bit host part.
    - Each company allocates one random prefix for its private address space.
    - 2 companies merge.
    - Probability for collision/renumbering ?

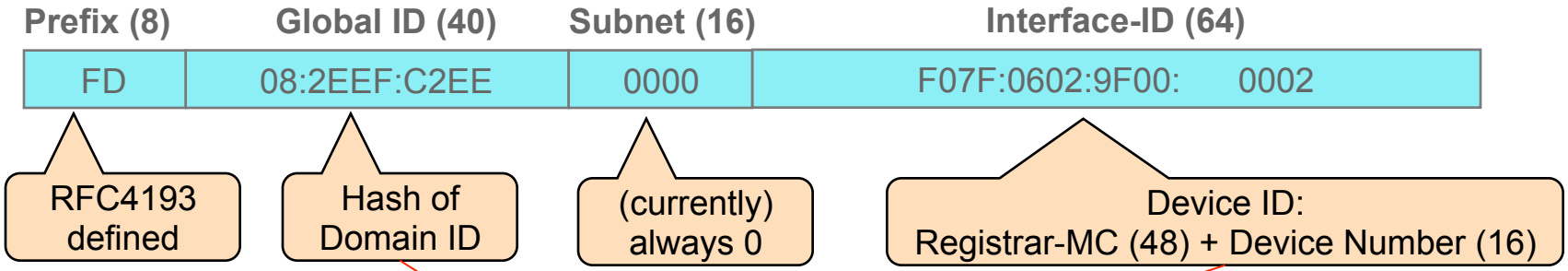
Unique Local Addresses (ULA) – RFC 4193

- Do not abuse
  - Only allocate one (at best few) prefixes
    - Eg: do not allocate ULA prefix per site (if you have many sites).
    - allocate subnets instead
  - Do not design solution for many prefixes to connect
    - high collision probability then!
  - See also: **draft-ietf-v6ops-ula-usage-considerations**
- Optionally, register prefix:
  - https://www.sixxs.net/tools/grh/ula/
- Use for routing in eg: ad-hoc network solutions
  - Hop-by-hop communication: Link-local
  - Network wide communication: ULA

ULA examples

- Back to my Mac
  - Ad-Hoc secure “VPN” solution in MacOS
  - https://tools.ietf.org/html/rfc6281
  - ULA as host identifier.
    - Survive network attachment (physical interface IPv6 address) change
    - ULA only easily useable solution – everything else would have required more code
- NEST
  - ULA between NEST equipment to not depend on or fail with other IPv6 addressing in the home network.
- Autonomic Networking
  - Secure inband management plane – indestructible VPN across network
  - draft-behringer-anima-autonomic-addressing-02.txt

Example ULA scheme: Autonomic Networking




```
Router#show autonomic device
 UDI                          PID:A901-6CZ-FT-A SN:CAT1651U08C
 Device ID                    f07f.0602.9f00-2
 Domain ID                    cisco.com
 Domain Certificate           (sub:) ou=cisco.com+serialNumber=
                              PID:A901-6CZ-FT-A SN:CAT1651U08C,
                              cn=f07f.0602.9f00-2
 Certificate Serial Number 04
 Device Address               FD08:2EEF:C2EE:0:F07F:602:9F00:2
        Domain Cert is Valid
```

Privacy addresses: RFC4971

- *How do you pay for Internet services ?*
  - *Taxes, Internet service cost, Advertisements, Equipment ?*
    - *Sure, but why stop there ?*
  - *Your Personal Data !!*

- Some history
  1. ISPs had tight IPv4 address space:
     - Dynamic allocated IPv4 address to residential subscribers when needed
  2. ISPs recognize value of static IPv4 address
     - Dynamic address for free (even if always on). Sell static address
  3. Users recognize value of dynamic addresses in copyright infringement lawsuits
  4. Privacy advocats recognize value of dynamic addresses
  5. IETF RFC
  6. In some countries SP provide dynamic IPv6 prefixes
     1. German IPv6 council 2012 recommends them:
     2. http://web.archive.org/web/20121207001716/http://www.ipv6council.de/documents/leitlinien_ipv6_und_datenschutz.html

everybody
On the Internet, ~~nobody~~ knows you're a dog

Source: https://en.wikipedia.org/wiki/On_the_Internet,_nobody_knows_you're_a_dog

Privacy addresses

- How does it work ?
  - Extensions to SLAAC
  - Well-known host part for “inbound” connections (EUI64)
  - Dynamically assigned “privacy” host part for “outbound”.
    - Change with every app/session/connection/…
- Effectiveness / benefits arguable
  - Depends on how many “people” can share the same prefix and assign addresses - Without being tracked. And in-country legal tracking obligations of SPs.
- Application issue: Changing Addresses and Cookies often don’t mix!

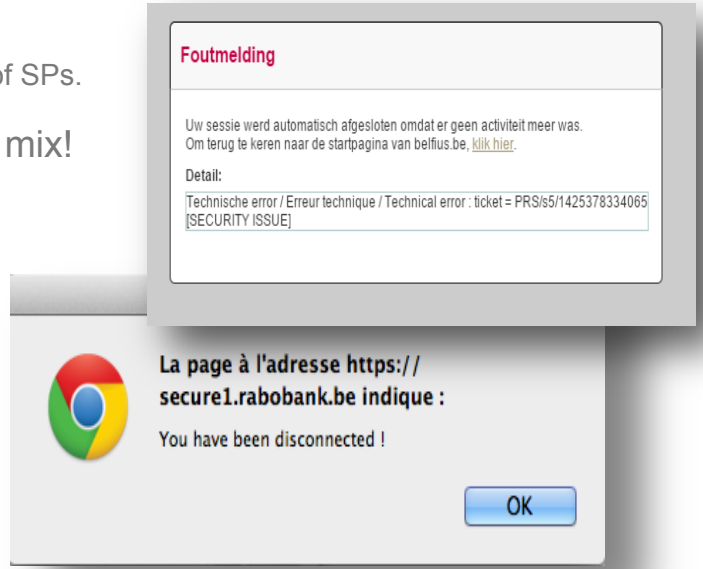
Source: wikimedia and Pinheiro

+

Source: http://www.woot.com/offers/tracker-bluetooth-tracking-device-2pk-13

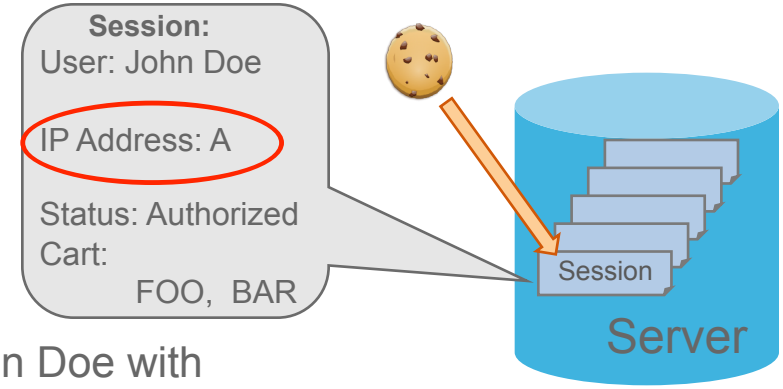
=

Source: Frustrated Belgian Cisco employee

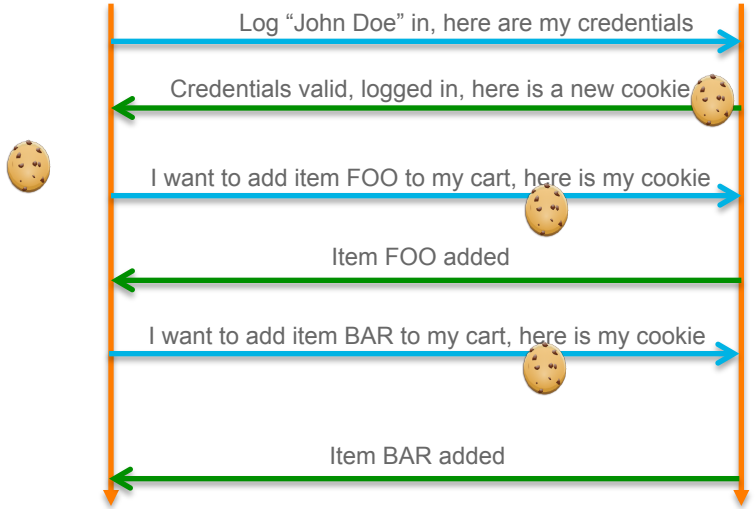


HTTP Session Cookie

- HTTP protocol:
  - No transaction/session concept
- HTTP Server (Application):
  - Keeps (user) ‘session’ state
  - HTTP transactions mapped to session via
    - **HTTP header cookie**
    - Usually encodes “index” into table of all open session
- Prohibit ‘session hijacking’:
  - Store client IP address
  - Check every HTTP transaction against it
  - *Because cookies can be snooped*
    - *Wire: HTTP, Client: Malicious Scripts*



John Doe with
IP address A



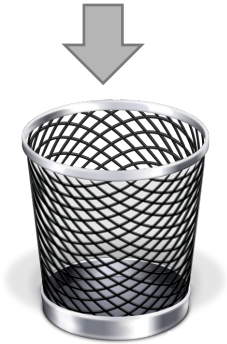
Post Mortem

- When client IP address changes:
  - IP address tracking with Cookie fails
  - Session reset / terminated
- Applies to all causes for address change, such as:
  - Privacy adddresses
  - Interface mobility (wired to wifi)
  - SPs doing dynamic address
  - Carrier Grade NAT (“CGN” – only when not RFC6888 compliant)
  - Happy Eyeballs (RFC6555 – change between IPv4/IPv6 address family)
- Impact:
  - At least two content providers in Belgium have stopped dual-stack deployment (2015)
    - Providers Infosec teams not ready to unlink session cookie from IP address
  - Slows down IPv6 content/services deployment

Highlander Prophecy

There Can Be Only One

IP address per user



Preventing Session Cookie Stealing

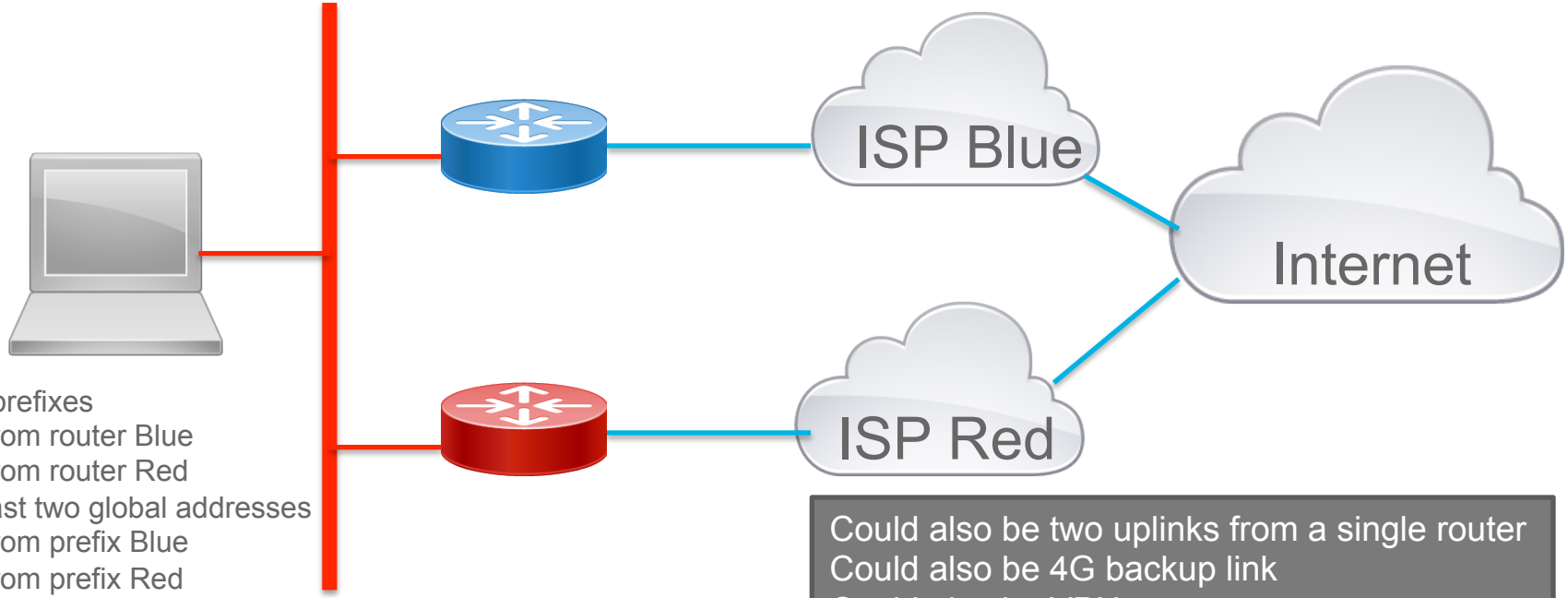
- Working with OWASP to establish BCP:
  - Open Web Application Security Project
- Checking IPv4 address mostly useless in CGN world
- Prevent cookie stealing on the path
  - Encrypt with HTTP2 or TLS. Eliminates network eavesdropping
- Prevent cookie stealing by hostile script
  - Add “HttpOnly” in Set-Cookie (prohibits cookie exposure to client-side scripts)
- Rely on MPTCP / SCTP
  - When adding or moving to new address, MPTCP/SCTP provide secure, seamless handover.
- References:
  - RFC6883 (sect 8.2), draft-vyncke-v6ops-happy-eyeballs-cookie

Multiple Addresses for better scaling & virtualization

- Q: What if you have a server that needs more than 64k connections ?
  - You run out of TCP/UDP port space
- A: Use multiple addresses
  - Each one has its own port range

- Q: What if you have multiple apps that want the same (well-known) port ?
  - Think each app has a “web-server” to interact/manage
  - Think apps will be containerized (eg: docker)
- A: Give each application/container a separate IPv6 address
  - And DNS name of course
  - DNS-SD can solve this issue too, but with containerization/virtualization, multiple addresses may actually be easier to manage (address per container).

Multi-homing addresses (eg: for Resiliency)

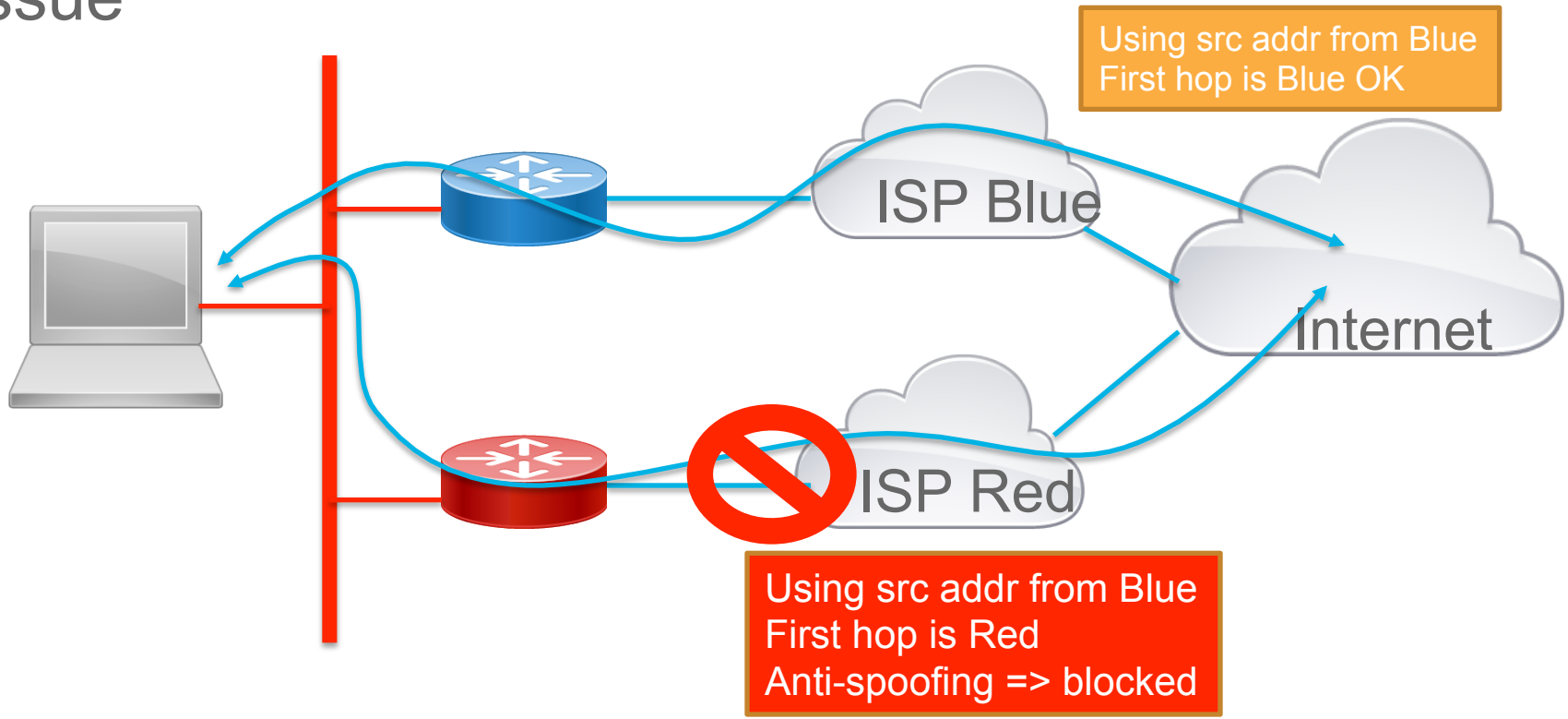


Two prefixes
1. From router Blue
2. From router Red

At least two global addresses
1. From prefix Blue
2. From prefix Red

Could also be two uplinks from a single router
Could also be 4G backup link
Could also be VPN
Or even multi-POP

Multi-homing addresses Issue



Multi-addressing issues

- Send packets with selected source address to correct first-hop router
  - Mainly a host network stack issue. Difficult to work around in app.
  - IETF work in progress.
- DNS lookup
  - Different “ISP” may have different views
    - Eg: SP-video servers only reachable via SP itself, and in DNS of that SP
  - Per-source-address DNS lookup ??
- What if you also have multiple interfaces (eg: Wired / WiFi) ?
  - Different addresses – but eg: all addresses from same ISP
  - No need to do multiple DNS lookups, distinguish source addresses
- What if ISP-edge routers not directly connected but via multiple hops ?
  - Homenet/Enterprise network with multi-ISP exit.

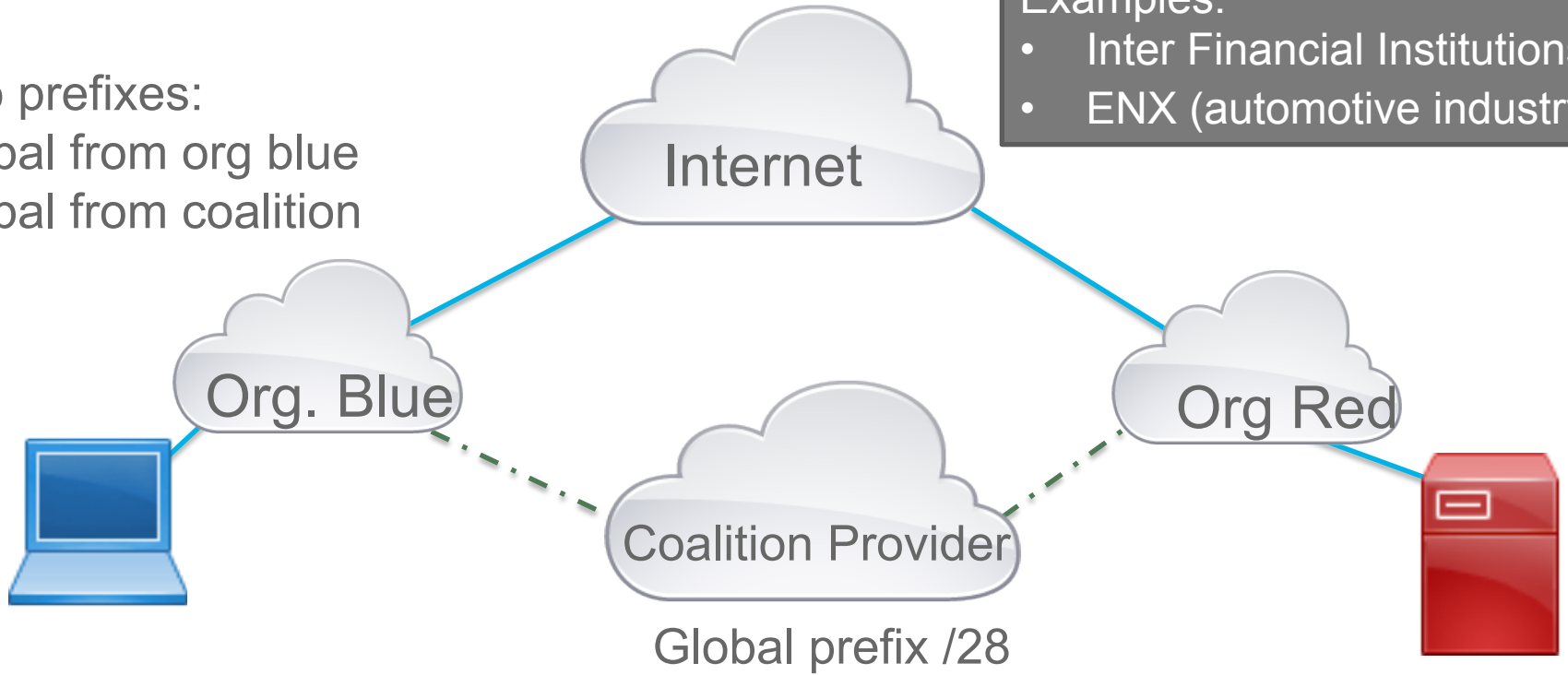
Multi-addressing solutions

- Provisioning Domain (PvD)
  - Name (string, eg: “ISPfoo”) associated with address/prefix
  - Permit to identify addresses belong to same PvD
  - Permit to enter PvD in app as “user-visible” policy choice
    - To select “best” service
  - Need support in multi-interface routers
  - IETF work in progress
- Source/Destination routing (in network, not in host)
  - Route inside home/enterprise based on combination of destination and source address
    - Eg: Both ISP routers route to all “Internet destinations”
    - If I use source-address from ISP red, I wand to route towards the red ISP edge router.
  - IETF work in progress

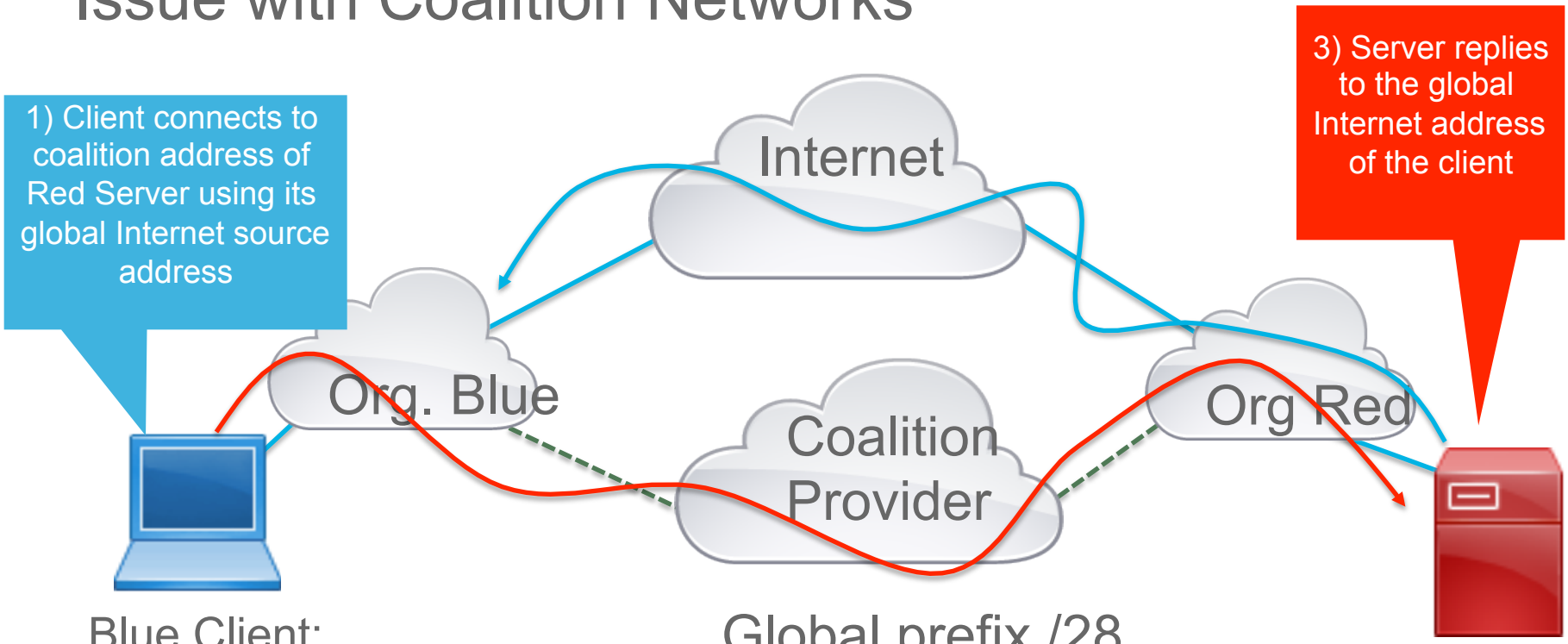
Coalition Networks

Two prefixes:
Global from org blue
Global from coalition

Examples:
- Inter Financial Institutions
- ENX (automotive industry)



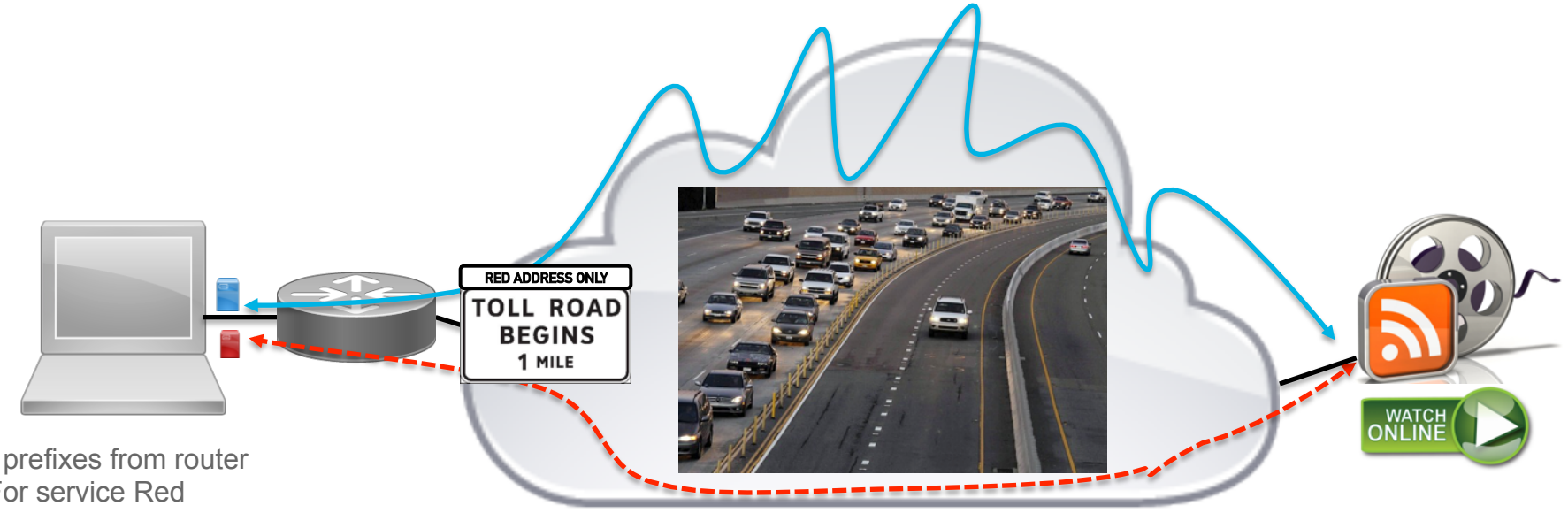
Issue with Coalition Networks



Issue with Coalition Network

- Again, the host must select the right source prefix
- Access router must ‘tag’ the prefixes
- IETF work in progress

Service Selection



Source: http://articles.latimes.com/2012/jul/01/local/la-me-toll-roads-20120701

Two prefixes from router
1. For service Red
2. For service Blue

At least two global addresses
1. From prefix Blue
2. From prefix Red

Traffic engineering
Different QoS
Different routing (€€€ or security)

"Pay more for HD during prime time"

Example: TeraStream

- https://ripe67.ripe.net/presentations/131-ripe2-2.pdf, slide 10
  - Next-gen network by Deutsche Telekom
- Service type indicated by bits in the address prefix
  - Compared to bits in host-part: better to manage by SP:
    - Routing table entries, ACLs, policies, …
- Simple devices (single service – TV, Phone,…) may just need one address
  - No advanced stack needed. Static or DHCP
    - DHCP needs to know service of device. Could be as simple as home-router-port-based
- “Interesting” case: multi-service devices
  - Service selection per “session” / “application” / …

IPv6 Nodes ~~can have~~ will have Multiple Addresses

PER INTERFACE – AND MAYBE MULTIPLE INTERFACES TOO

One Link-Local Address

Zero or more Global Addresses

Assignment – that is the easy part:

- Through DHCPv6 which can give multiple addresses
- Through Stateless Address Auto Configuration (SLAAC)
  - Based on several distinct Router Advertisements from one or more adjacent IPv6 routers
  - Each Router Advertisements can include multiple /64 prefixes
  - Nodes then generate 1, 2, ... Addresses per prefixes
- Through Novel schemes – manual/application/DNS based easily possible
  - Just ensure uniqueness of addresses and “routeability”

- Selection – much more tricky

Today: “limited” source/dest addr selection

- RFC6724 (superceeds RFC3484)
- Considers set of available local (source) and remote peer (destination) addresses
- Create sort-list of combinations to try
- Tries to eliminate unreachable destinations
- Tries to match common longest prefix between available source and destinations – longer == try earlier
- Cache recent results
- No “policy” aspects or PVDs considered
- Implementations in different OSs vary, BUT COULD ALSO DO THIS/ ENHANCE THIS IN THE APPLICATION.

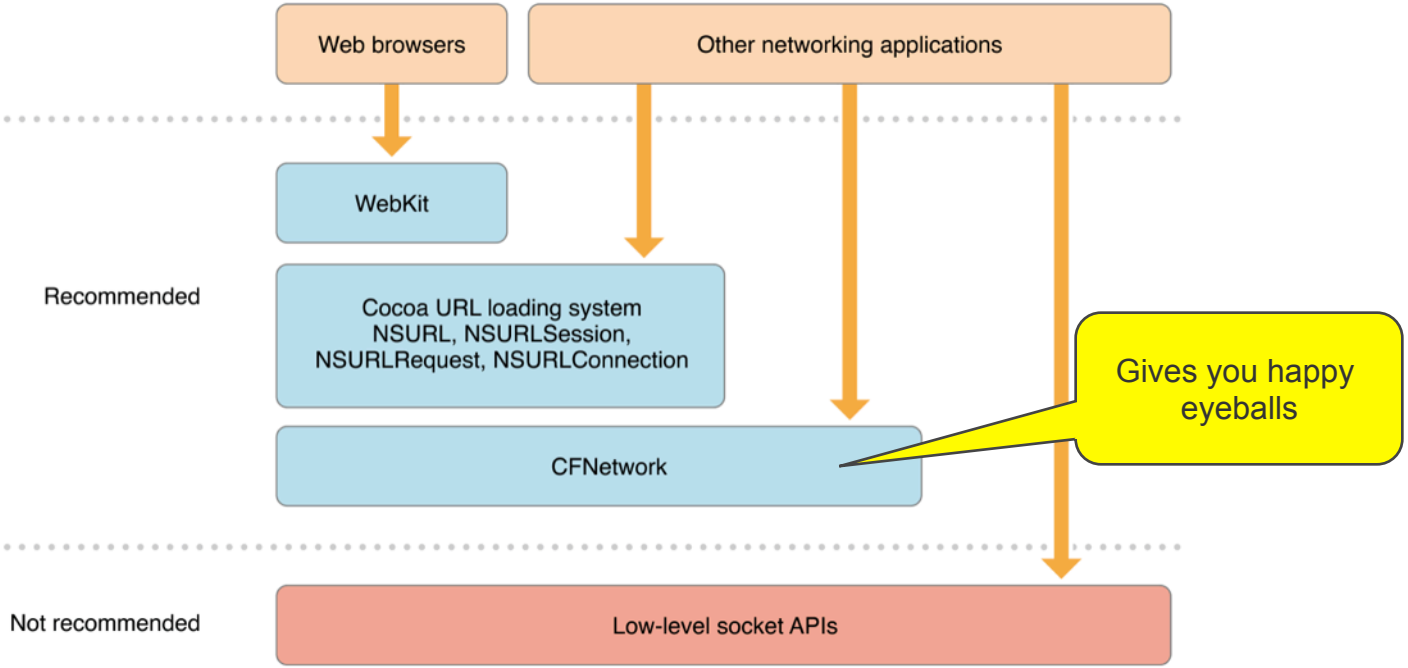
Source Address Selection: RFC6724 (nee RFC3484)

- Candidate set of addresses
  - From egress interface
  - Sorted list
- 7 rules
  - Prefer same address
  - Prefer appropriate scope
  - Avoid deprecated addresses
  - Prefer outgoing interface
  - Prefer matching label
  - Prefer temporary addresses
  - Use longest matching prefix

Today: Making eyeballs more and more happy

- Original “Happy Eyeballs” RFC6555
  - Procedures for dual-stack, IPv4 vs. IPv6 address selection
  - “Prefer IPv6”. Eg:
    - https://www.ietf.org/mail-archive/web/v6ops/current/msg22455.html
- Term is also applied to other selections
  - Not necessarily (yet) with equivalent documentation
  - IPv6 service (source/dest) address selection
  - Transport protocol / parameter selection (eg: TCP/SCTP/…)

Happy Eyeballs – beyond dual-stack
Apple – MacOS/iOS



Today: Interactive Connection Establishment (ICE)

- Originally thought for NAT traversal for P2P apps (IPv4)
  - Telephony/Conferencing – UDP centric
  - Connectivity only possible when initiated from inside. If both sides do this, then connection setup is complicated (ping from both sides).
  - If direct connectivity impossible, use “TURN” servers as relays.
- Many pieces applicable without NAT or with TCP (instead of UDP)
  - When firewalls are in the path (with also inside-2-outside pinholes)
  - When multiple (IPv6) addresses are present
- ICE libraries can be a good starting point for service selection.
  - And maybe necessary anyhow (before IPv4/NAT dies totally)
  - https://github.com/cisco/NATTools

Yes, that's right. Choose your source address, I'll make sure packets go down the right path with the right QoS and right filtering.

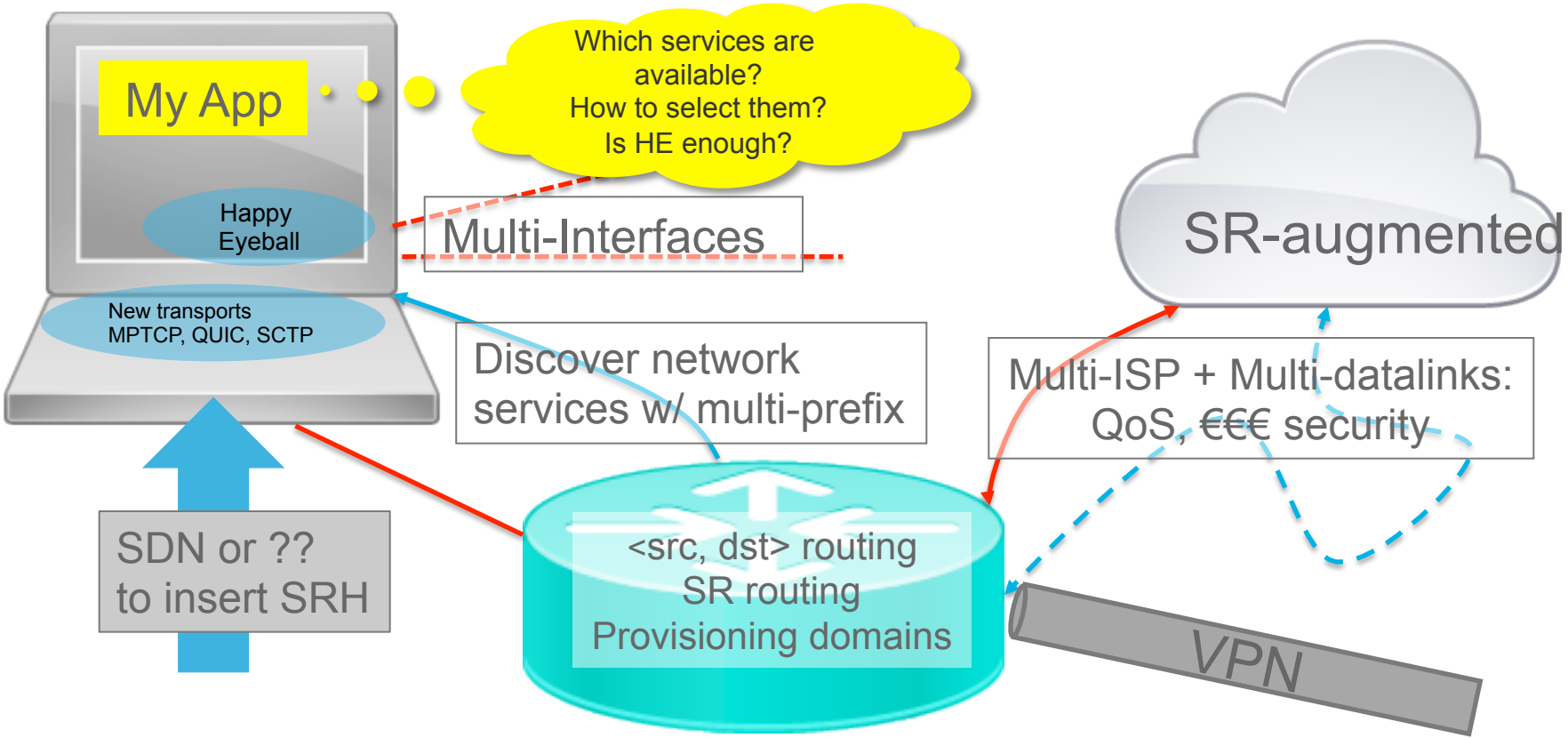
"So, the source address I select affects the path and associated policy throughout the network?"

Yes, that's right. Choose your source address, I'll make sure packets go down the right path with the right QoS and right filtering.

Yikes! What do I do! I've never asked the user for this kind of information before!

Credit: Mark Townsley

API to Select New Services Enabled by IPv6



From Services Discovery to Service Selection

- IPv6-centric networks are versatile
  - Multi-interfaces with varying €€€, security, network properties
  - Each network can deliver multiple prefixes (SR, ...)
- Network services advertised with multi-prefixes in addition to MIF provisioning domains
- Multi-prefixes require <src, dst> routing (BCP 38) in the network (homenet? CPE?)
  - Multiple ways of doing it
- NEW transport protocols MPTCP, SCTP, QUIC with different services
  - Also impacting network/security devices
- Happy Eyeball extensions not efficient and too limited in selecting services
- => Apps need API to
  - Discover the services
  - (Present them to the users)
  - Select the preferred one

How to develop for all these addressing options ?

- Basic socket API across OS will have
  - Some (subset)/variant of RFC6724(source,dest) address selection
  - And maybe some happy eyeball policies
- Most everything else is very OS dependent and/or via libraries/SDK
  - And fragmented across OS, variety of libraries
- Coalescing advanced transport layer functions into a simple re-useable, open source stack is challenging
  - No history of cross-OS integrated transport functions to make life easier
  - Most transport functions started in OS kernels
    - Internet architecture: per-app de-multiplexing at transport layer
  - Recent work more and more at userland:
    - Browser transport code “TCP/SCTP over UDP”,QUIC (Google), .. RTCweb,…

Example project: neat

A New, Evolutive API and Transport-Layer Architecture for the Internet

- European Union Horizon 2020 project
  - https://www.neat-project.org
- Develop architecture – and (as much as feasible) open source implementation of host side transport stack to simplify rich service selection transport functionality.
  - Architecture:
  - https://www.neat-project.org/wp-content/uploads/2016/02/D1.1.pdf
- Started/run primarily by participants in IETF network, transport & app. areas.
- Plan to include policy (metadata attributes) at API level to appropriate underlying mechanisms – such as service-selection via address selection.

Conclusion

Takeaways

- Making apps work with IPv6 as well as IPv4 is not difficult
  - Dual addressing support is also easy when using AF independent APIs
- There is a lot more possible with IPv6 than IPv4 (addressing)
  - Some of it will still be difficult to program
- Use high-level APIs when you can
  - And create them if you can't !
- Use names and DNS!
- Make sure your app (media & signaling) can work with ONLY IPv6.
- Avoid u32
- Test, test, test !
  - Don't be afraid to engage users !

Thank you

We’re ready. Are you?