Networking for the Modern Internet

Communicate your app's needs to the networking layers

Session 714

Stuart Cheshire Apple DEST

Topics

Update on ECN (Explicit Congestion Notification)

IPv6 and your applications

International text in networking

Cellular versus Wi-Fi

Network Quality of Service (QoS)

ECN Update

Explicit Congestion Notification

Recap from WWDC 2015

SQM (Smart Queue Management)

ECN (Explicit Congestion Notification)

Reduces delays and retransmissions

See Your App and Next Generation Networks from WWDC 2015

Your App and Next Generation Networks WWDC 2015

---

iOS 9 Launch Revealed Problem in Germany

One German ISP marked all packets "Congestion Experienced"

- Affected VPN connections
- Fixed by German ISP within a couple of weeks

No other problems reported anywhere else in the world

The Internet is now safe for ECN

Ramping Up Usage of ECN

In iOS 9.3 and OS X El Capitan v10.11.5, 5% of outgoing connections now request ECN

In Developer Seed, 100% of connections request ECN on Wi-Fi and three selected carriers

T··Mobile·

Three.com.hk

TELSTRA

Web Sites Supporting ECN

September 2014: Alexa top million web sites supporting ECN: 56%

- Enabling Internet-Wide Deployment of Explicit Congestion Notification

June 2016: Alexa top million web sites supporting ECN: 70%

- http://ecn.ethz.ch/

June 2016: Alexa top million (IPv6 only) supporting ECN: 83%

Time to Start Doing ECN Marking

Mark packets instead of dropping

- Reduce packet loss
- Reduce delays and wasted bandwidth due to retransmissions
- Better user experience
- More efficient use of network

IPv6 and Your Applications

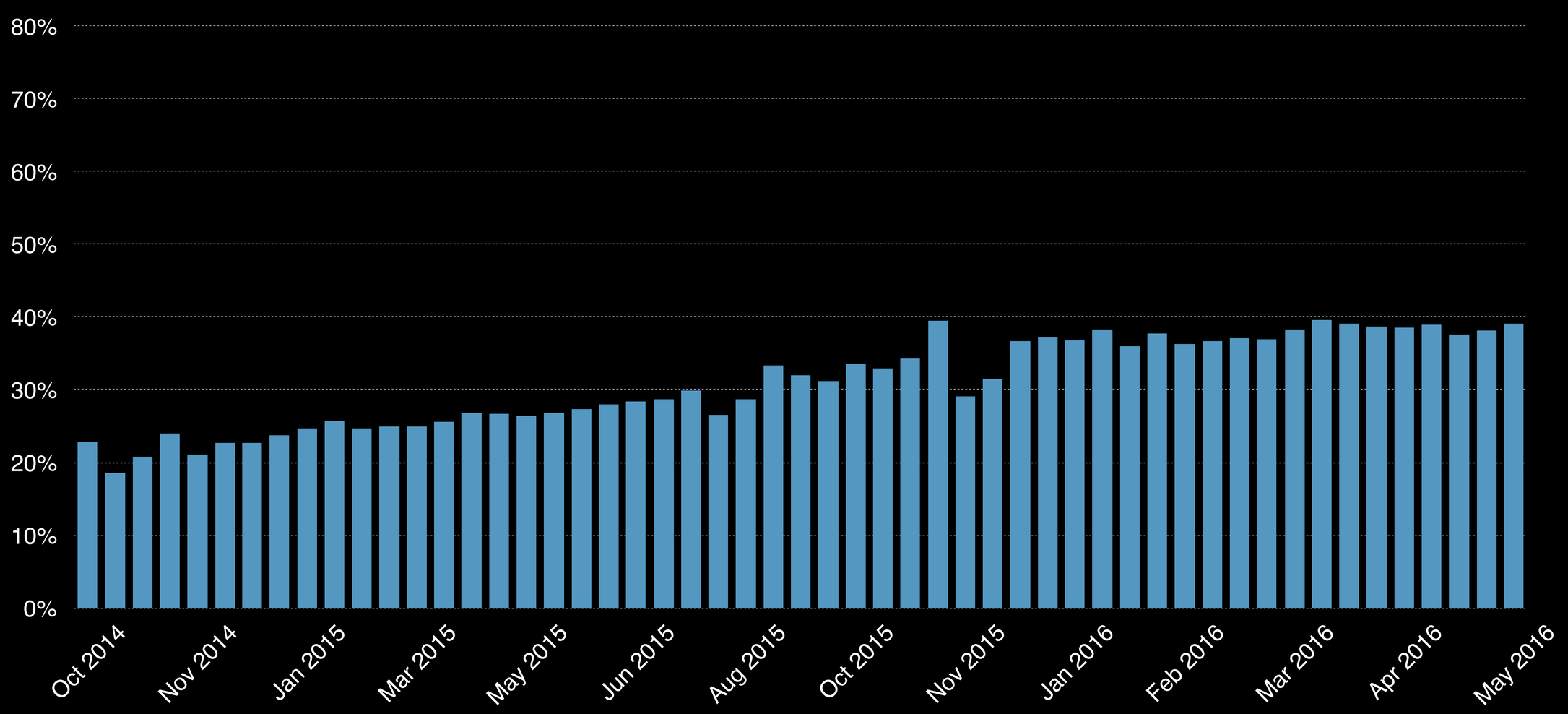
World IPv6 Launch 4 Years Ago Last Week

6/6/2012

IPv6 Continues to Grow

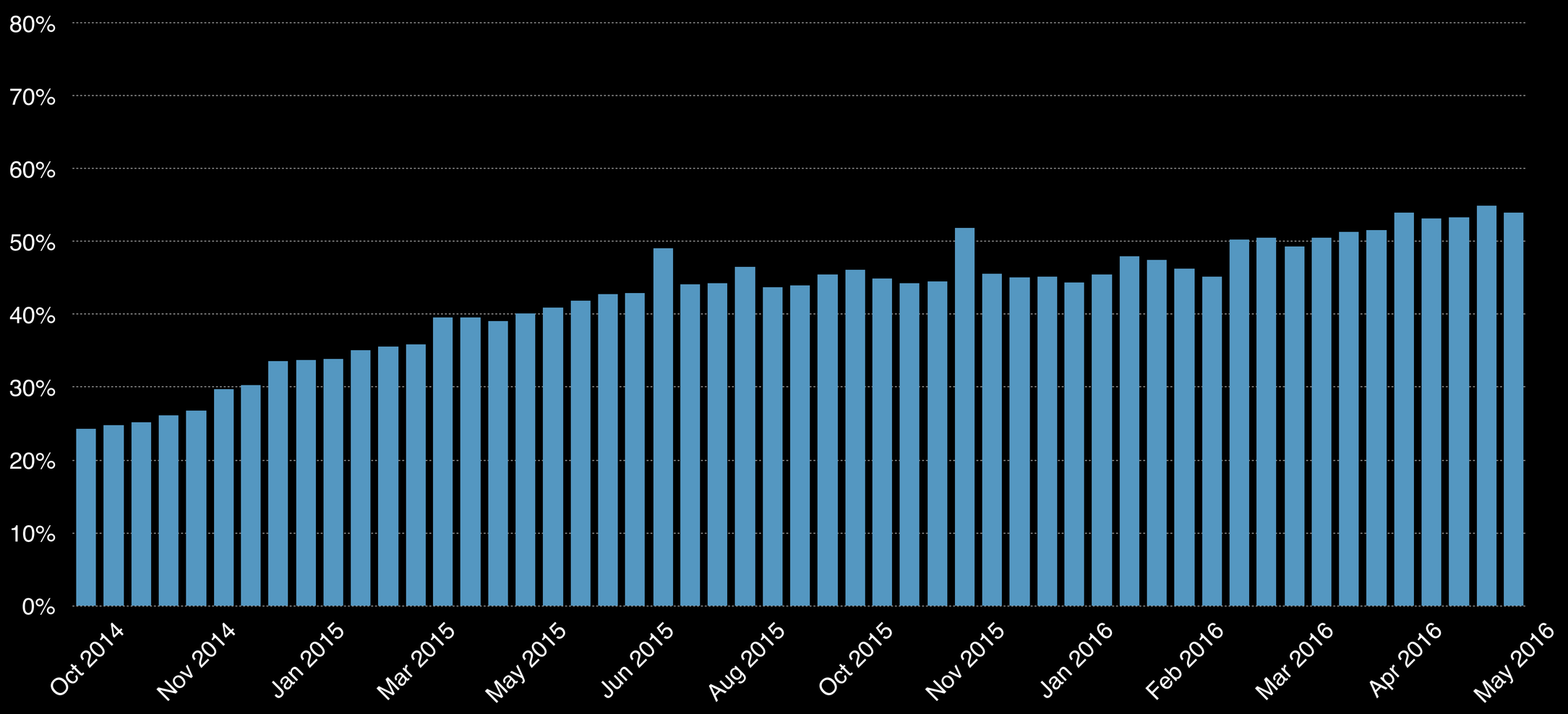
Access to www.apple.com over IPv6

In Belgium



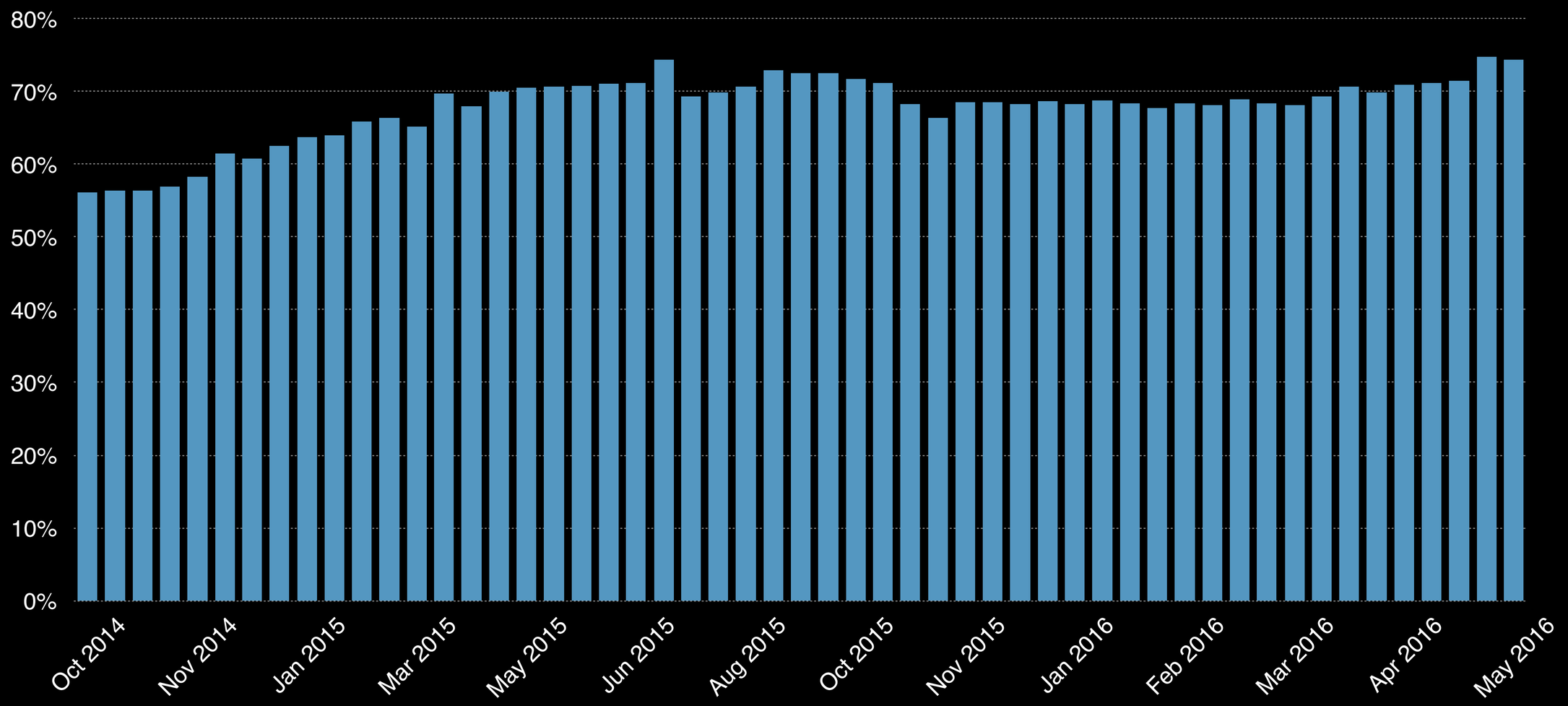
Access to www.apple.com over IPv6

On T-Mobile USA



Access to www.apple.com over IPv6

On Verizon Wireless



Better for Carriers

IPv6 Benefits for Mobile

LinkedIn

10% IPv6

10–40% faster than IPv4

Source: Zaid Ali Kahn, Senior Director, Global Infrastructure Architecture and Strategy at LinkedIn

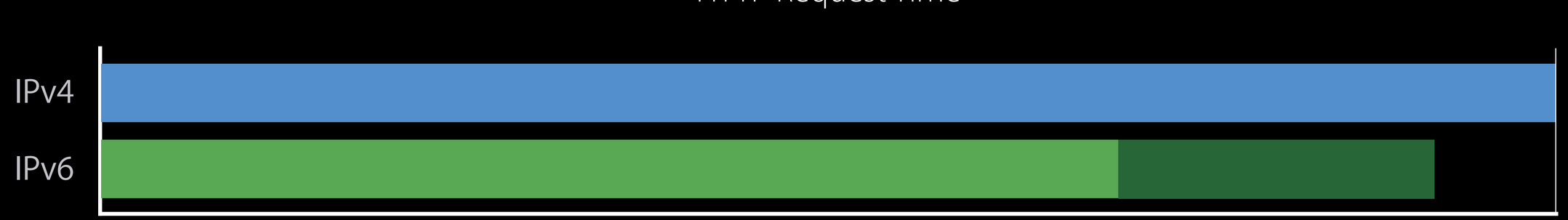
- Cisco Ecole Polytechnique Networking Innovation and Research symposium, March 2016
- https://www.linkedin.com/pulse/ipv6-measurements-zaid-ali-kahn
- https://www.youtube.com/watch?v=FUtG89C8h_A

IPv6 Benefits for Mobile

Facebook

45% IPv6

15%–30% faster than IPv4



Source: Paul Saab, Facebook engineer

- Networking @Scale, May 2016
- https://code.facebook.com/posts/1192894270727351/ipv6-it-s-time-to-get-on-board/
- https://code.facebook.com/posts/1036362693099725/networking-scale-may-2016-recap/

Better for Users

Supporting IPv6-Only Networks

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 watching Your App and Next Generation Networks.

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

No Detectable Change in App Acceptance Rate

What To Do if Your App Was Rejected

Test your app for yourself on your own NAT64 network

- Review Your App and Next Generation Networks presentation from WWDC 2015
- Test here on WWDC NAT64 network and come talk to us at the WWDC labs

Use address-family agnostic APIs

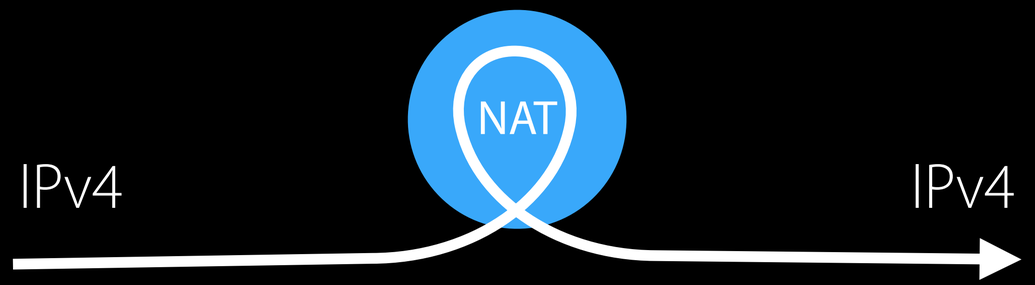
- Use higher-layer Foundation APIs like NSURLSession and CFNetwork
- Avoid low-level BSD sockets and third-party networking libraries, which lack these capabilities

Use hostnames, not literal addresses

Your App and Next Generation Networks	WWDC 2015

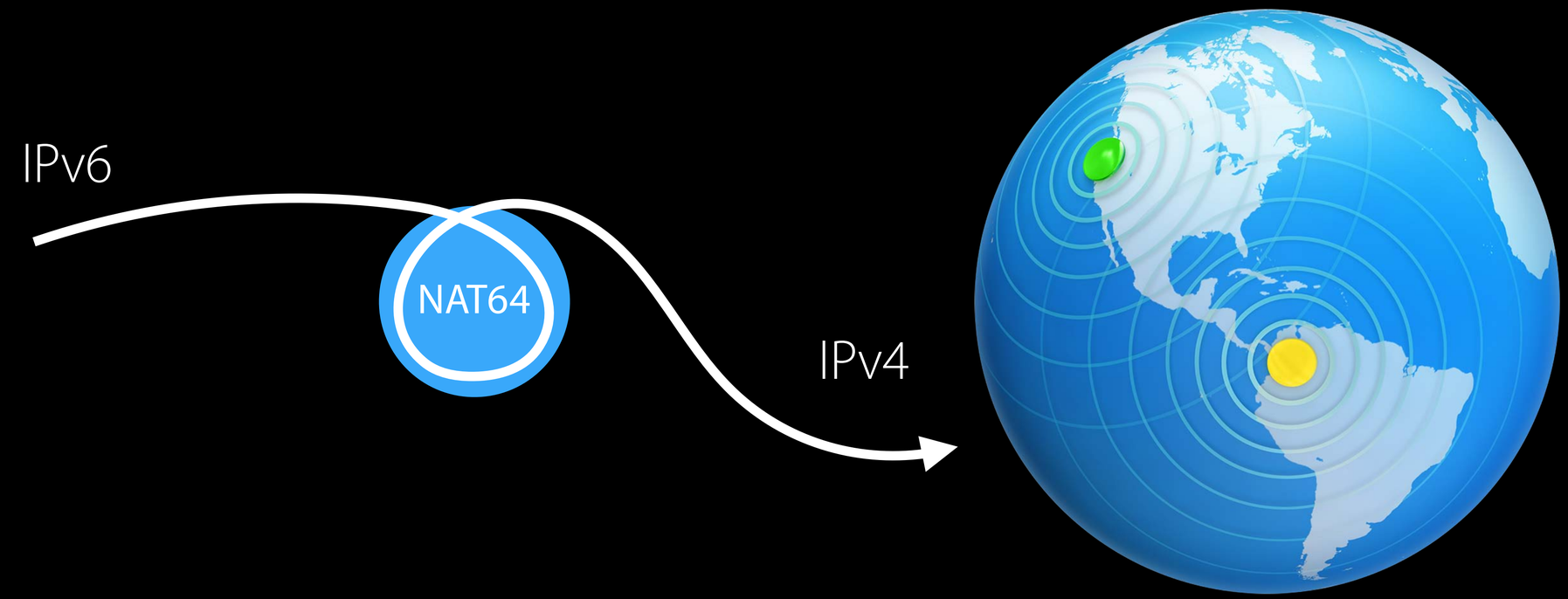
IPv4-Only Server

Client on IPv4-only network



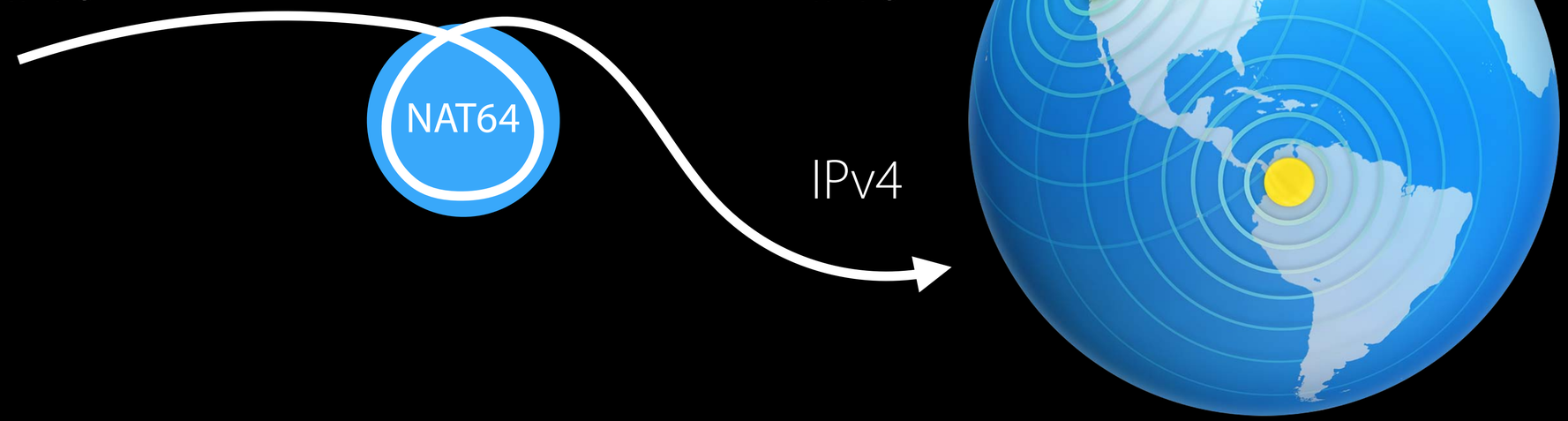
IPv4-Only Server

Client on IPv6 + NAT64 network



Dual-Stack Server

Client on IPv6 + NAT64 network, using literal IPv4 address



Dual-Stack Server

Client on IPv6 + NAT64 network, using hostname

IPv6

NAT64

IPv6

IPv4

Using Literal IPv4 Addresses

Literal IPv4 addresses supported in selected APIs

- High-level APIs like NSURLSession and CFSocketStream
- The `getaddrinfo()` call, for low-level APIs like BSD sockets
  - Need to use `getaddrinfo()` if using UDP

Using literal IPv4 addresses will prevent direct IPv6 connection to a dual-stack server

```
// Using getaddrinfo() with Literal IPv4 Addresses

struct addrinfo hints = {
    .ai_family   = PF_UNSPEC,
    .ai_socktype = SOCK_STREAM,
    .ai_flags    = AI_DEFAULT
};

struct addrinfo *res0;
getaddrinfo("192.0.2.1", "https", &hints, &res0);  // Error checking omitted for brevity!

for (struct addrinfo *res = res0; res; res = res->ai_next) {
    int s = socket(res->ai_family, res->ai_socktype, res->ai_protocol);
    connect(s, res->ai_addr, res->ai_addrlen);  // More error checking omitted!
    // Do some stuff ...
}

freeaddrinfo(res0);
```

Connecting to Devices on the Local Link

Ideally, devices should support IPv6

If not, alternative is for device to support IPv4 link-local (RFC 3927)

If device doesn't support IPv6 and can't do IPv4 link-local:

- Inform App Review when you submit your app
- This is not grounds for rejection
- Probably is grounds for putting one of these on the device →

All off-link communication from your app must still be compatible with IPv6 and NAT64

Legacy IP logo courtesy of Phil Benchoff at Virginia Tech

IPv6 Best Practices

Support IPv4 and IPv6 end to end

- Address-family agnostic clients
- Dual-stack servers

Use names, not addresses

- Lets DNS64 work
- Lets clients connect directly to dual-stack servers

Using literal IPv4 addresses

- Works in selected APIs
- Prevents direct IPv6 communication to a dual-stack server

International Text in Networking

International Text in Networking

Latin Alphabet (Polish)	Małgorzata@example.club
Greek Alphabet	δοκιμή@παράδειγμα.δοκιμή
Traditional Chinese Characters	我買@屋企.香港
Japanese Characters	甲斐@黒川.日本
Cyrillic Characters	чебурашка@ящик-с-апельсинами.рф

International Text in Networking

Some Unicode Terminology

Unicode A set of integer code points in the range 1 – 1,114,111 (1 – 0x10FFFF) where each code point represents (with some exceptions) a human-meaningful visual "character"

UTF-32 Each Unicode integer code point stored using a single 32-bit integer (so endianness matters)

UTF-16 Each Unicode integer code point encoded using one or two 16-bit integers (so endianness matters)

UTF-8 Each Unicode integer code point encoded using one to four 8-bit integers in a specified order (so no endianness problems)

UTF-8 History

Designed by Ken Thompson on a placemat in a New Jersey diner one night in 1992

Source: Rob Pike https://www.cl.cam.ac.uk/~mgk25/unicode.html#history

UTF-8 Syntax

Code points 0x00 – 0x7F same as ASCII

- Code points 0x00 – 0x7F encoded using octet values 0x00 – 0x7F
- So all current 7-bit ASCII files are also valid UTF-8
  - With the same meaning

Higher code points use multi-octet sequences

- Multi-octet sequences use octet values 0x80 – 0xF4
- Existing files already assigning other meanings to octet values 0x80 - 0xFF (e.g. ISO 8859-1) are not automatically compatible

UTF-8 Multi-Octet Sequences

Single octet ASCII character
(Code points 1–127)

0XXXXXXX

First octet of
2,3,4-octet sequences

110XXXXX

1110XXXX

11110XXX

Continuation octets of
multi-octet sequences

10XXXXXX

UTF-8 Multi-Octet Sequences

00000 – 0007F	0XXXXXXX			
00080 – 007FF	110XXXXX	10XXXXXX		
00800 – 0FFFF	1110XXXX	10XXXXXX	10XXXXXX	
10000 – and up	11110XXX	10XXXXXX	10XXXXXX	10XXXXXX

UTF-8 Properties

No mid-string zero octets

Stateless character boundary detection

- Robust to insertions, deletions, errors, etc.

Strong heuristic detection

- e.g., any solitary octet with top bit set signals text as not valid UTF-8

Byte-wise, sorts same order as raw Unicode

IETF Policy on Character Sets and Languages

RFC 2277, January 1998

Protocols MUST be able to use the UTF-8 charset

Percentage of UTF-8 Web Pages

February 2012

80%

Source: Google

June 2016

87%

Source: W3Techs

The W3C strongly recommends that content authors should **only** use the UTF-8 encoding for their documents.

- Source: W3C: Who uses Unicode?

Punycode

Used for IDNs (Internationalized Domain Names)

A method of encoding a string of Unicode integer code points using only the following octet values:

- 0x61 – 0x7A
- 0x30 – 0x39
- 0x2D

i.e., octet values that, if (mis)interpreted as US ASCII, correspond to the following US ASCII characters:

- Letters a – z
- Digits 0 – 9
- Hyphen

Punycode

Example

78 6E 2D 2D 6F 6E 71 75 78 6B 31 68 6F 39 73 71

75 79 32 67 61 72 31 35 72 2E 78 6E 2D 2D 75 63

30 61 74 76 2E 78 6E 2D 2D 6A 36 77 31 39 33 67

ASCII

Punycode

xn--onquxk1ho9squy2gar15r.xn--uc0atv.xn--j6w193g

相信零可以成真.組織.香港

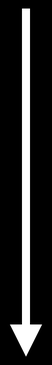
UTF-8

Comparison

E7 9B B8 E4 BF A1 E9 9B B6 E5 8F AF E4 BB A5 E6

88 90 E7 9C 9F 2E E7 B5 84 E7 B9 94 2E E9 A6 99

E6 B8 AF



UTF-8

相信零可以成真.組織.香港

Punycode

Automatically supported in Bonjour and DNS APIs

UTF-8 input

iOS 9 and OS X El Capitan

```
% ping 相信零可以成真.組織.香港
ping: cannot resolve 相信零可以成真.組織.香港: Unknown host
```

But they didn’t put the name into the DNS as UTF-8

Punycode

Automatically supported in Bonjour and DNS APIs

iOS 9 and OS X El Capitan

```
% ping 相信零可以成真.組織.香港
ping: cannot resolve 相信零可以成真.組織.香港: Unknown host
```

iOS 10 and macOS Sierra

UTF-8 input

```
% ping 相信零可以成真.組織.香港
ping xn--onquxk1ho9squy2gar15r.xn--uc0atv.xn--j6w193g (118.143.31.90): 56 data bytes
```

UTF-8 automatically converted to Punycode encoding
(and then (mis)displayed as if it were ASCII)

Punycode

Automatically supported in Bonjour and DNS APIs

Punycode is quite restrictive

- Doesn't support spaces—e.g., "Living Room Apple TV"
- Need to use UTF-8 for that

Bonjour and DNS APIs decide automatically

- Will try UTF-8 first
- If that fails, converts to Punycode and tries again
- Algorithm described in RFC 6763

Supports both rich-text UTF-8 Bonjour names and Punycode-encoded names

Email Addresses

On sign-up forms in apps on on the web, don't try to validate email address input

Accept what the user enters

Only reasonable restriction is that email address needs an @ sign

Send validation email to confirm address is "live"

Read if you're writing an email client or server

Internationalized Domain Names

Framework	https://tools.ietf.org/html/rfc5890
IDNA Protocol	https://tools.ietf.org/html/rfc5891
Unicode Code Points	https://tools.ietf.org/html/rfc5892
Right-To-Left Scripts	https://tools.ietf.org/html/rfc5893

Email Address Internationalization

Framework	https://tools.ietf.org/html/rfc6530
SMTP Extension	https://tools.ietf.org/html/rfc6531
Email Headers	https://tools.ietf.org/html/rfc6532
Delivery Status and Disposition Notification	https://tools.ietf.org/html/rfc6533
IMAP Support for UTF-8	https://tools.ietf.org/html/rfc6855
POP3 Support for UTF-8	https://tools.ietf.org/html/rfc6856

International Text Best Practices

UTF-8 is the new ASCII

Use UTF-8 for everything

Don’t worry about Punycode

Be liberal about what strings you accept

Cellular versus Wi-Fi

Wi-Fi Assist is your friend

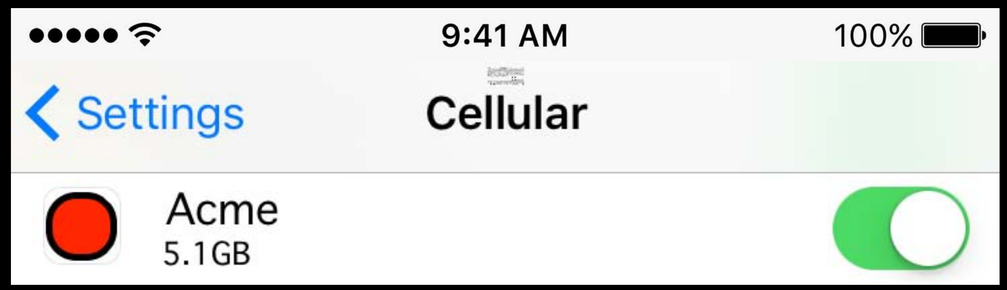
Wi-Fi Assist

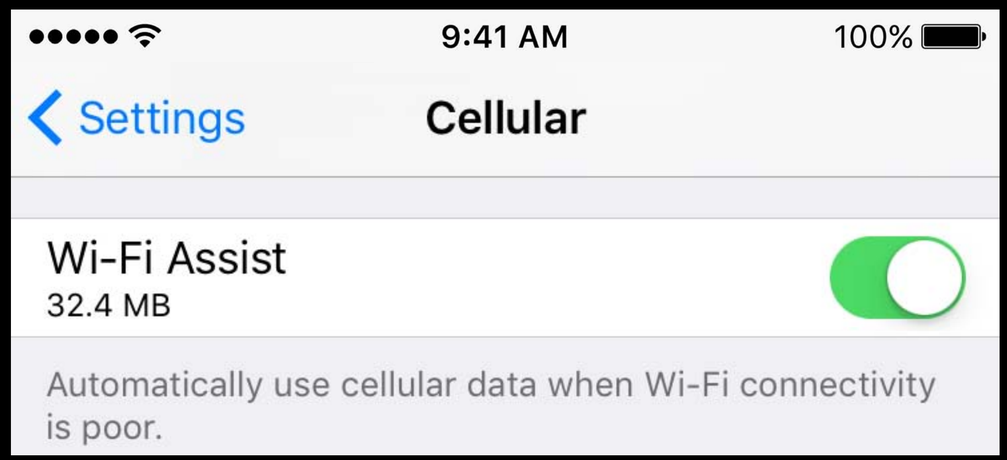
Wi-Fi

Cell

Express Intent — Control Cellular Networking

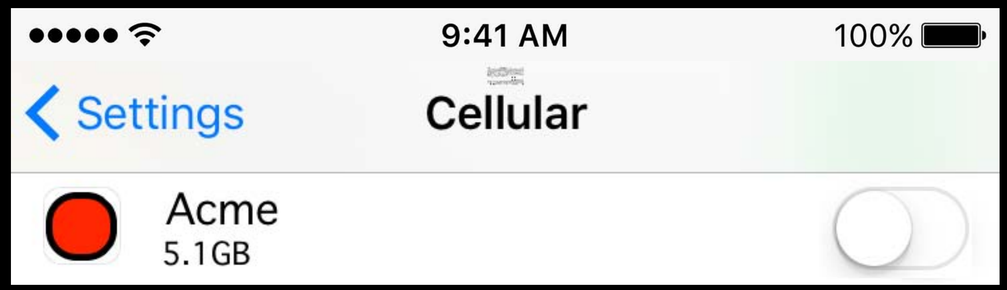
Global per-application control





Express Intent — Control Cellular Networking

Global per-application control



Per-Connection Control

Preflight checks can be misleading

```
SCNetworkReachabilityGetFlags(r, &flags)
let isReachable = flags.contains(.reachable)
let isCell      = flags.contains(.iswwan)
if isReachable && !isCell {
    // Should go over WiFi, but does it really?
    dataTask?.resume()
}
```

Per-Connection Control

NEW

Express Intent to control cellular data usage

1. Perform a network download/upload without preflight checks

Per-Connection Control

NEW

Express Intent to control cellular data usage

1. Perform a network download/upload without preflight checks
2. If session may be data intensive, request no cellular usage

- CoreMedia API

```
var asset: AVURLAsset = AVURLAsset(url: contentURL,
   options:[AVURLAssetAllowsCellularAccessKey: false])
```

- NSURLSession API

```
let configuration = NSURLSessionConfiguration.defaultSessionConfiguration()
configuration.allowsCellularAccess = false
let session = NSURLSession(configuration: configuration, delegate: self, delegateQueue: nil)
```

Per-Connection Control

NEW

Express Intent to control cellular data usage

1. Perform a network download/upload without preflight checks
2. If session may be data intensive, request no cellular usage
3. Should the session fail

Ask if user wants to use mobile data

or… just wait

Per-Connection Control

NEW

Express Intent to control cellular data usage

1. Perform a network download/upload without preflight checks
2. If session may be data intensive, request no cellular usage
3. Should the session fail
4. Continuously listen to better route events and repeat 1, 2, 3 (subject to app context)

```
func urlSession(session: NSURLSession, betterRouteDiscoveredFor
   streamTask: NSURLSessionStreamTask) {
   // Good news: WiFi associated once again!
}
```

Interface Selection Best Practices

Don’t assume that if you’re “on Wi-Fi” now your next connection will also be “on Wi-Fi”

- Network conditions change second to second

Express what you want to the networking layers

- Don’t just hope for the best

Networking Quality of Service (QoS)

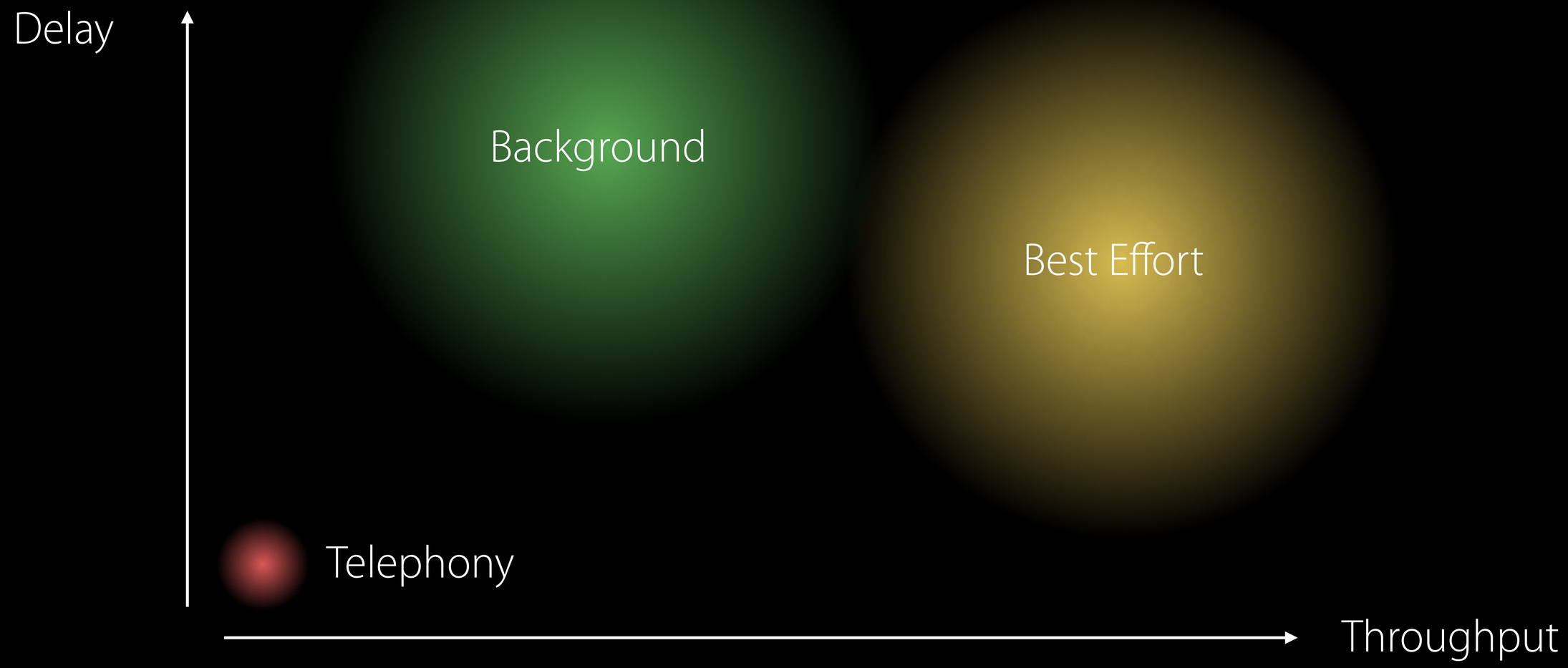
Network Service Type

In August of 2015, Apple and Cisco announced a partnership
to create a fast lane for iOS business apps.

With iOS 10 we are introducing new Quality of Service features
to optimize enterprise iOS apps with Cisco networks.

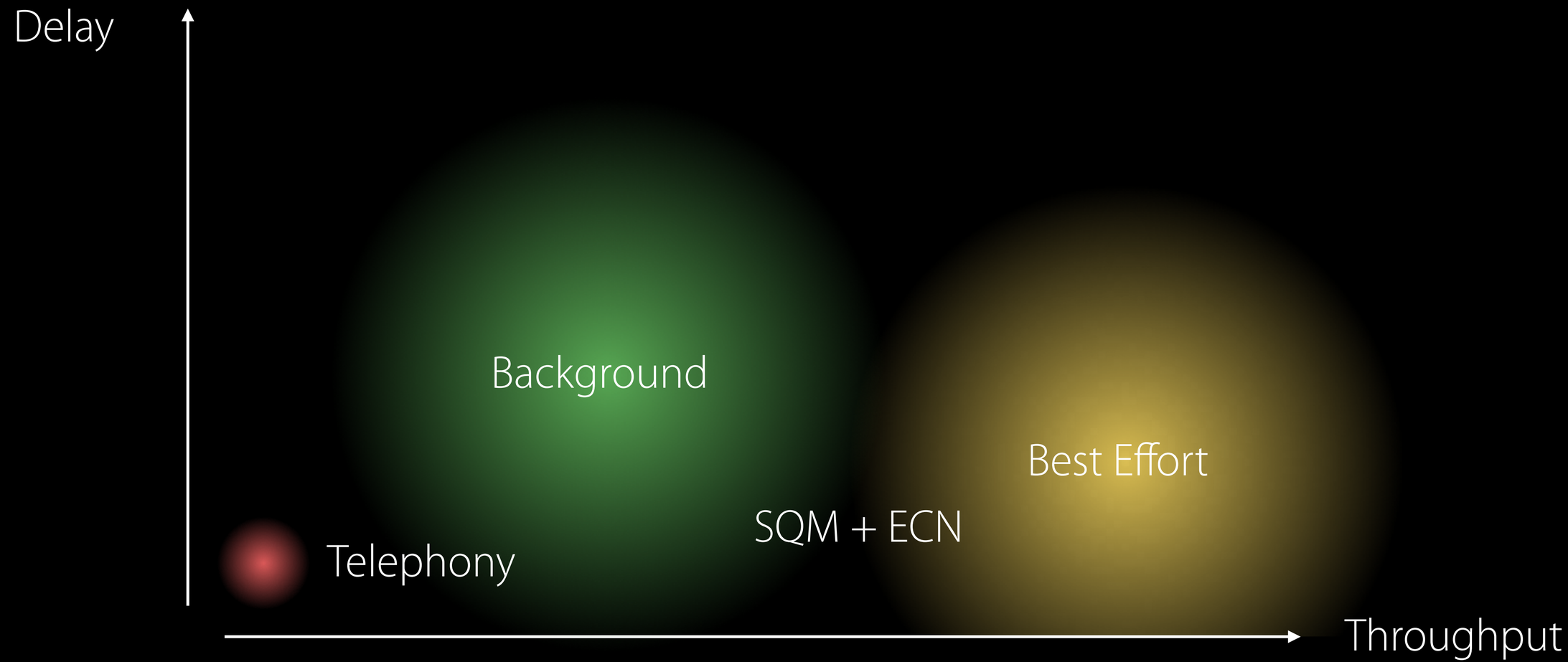
Network Service Types

Spectrum of characteristics



Network Service Types

Spectrum of characteristics



Network Service Type API

NEW

NSURLSession and CFNetwork

- Network Service Types
- Available in iOS 5, OS X 10.7, and later

Socket option to select the Network Service Type

- SO_NET_SERVICE_TYPE

Don’t try to use old IP Type Of Service (TOS) bits

- Incompatible interpretation between different Wi-Fi driver vendors
- No consistent interpretation on the network

NSURLSession and CFNetwork Service Types

```
stream.setProperty(NSStreamNetworkServiceTypeVoice,      forKey: NSStreamNetworkServiceType)
stream.setProperty(NSStreamNetworkServiceTypeVideo,      forKey: NSStreamNetworkServiceType)
stream.setProperty(NSStreamNetworkServiceTypeBackground, forKey: NSStreamNetworkServiceType)
```

Network Service Types Socket Option

```
int st = NET_SERVICE_TYPE_BK_SYS;
setsockopt(socketfd, SOL_SOCKET, SO_NET_SERVICE_TYPE, (void *)&st, sizeof(st));


// NET_SERVICE_TYPE_BE      Best effort
// NET_SERVICE_TYPE_BK_SYS  Background system initiated
// NET_SERVICE_TYPE_VI      Interactive Video
// NET_SERVICE_TYPE_VO      Interactive Voice
```

Link-Layer QoS Marking

Controls packet queuing and scheduling on network interface

For Wi-Fi also selects the WMM (Wireless Multimedia) Access Category

- AC_BK Background
- AC_BE Best Effort (default)
- AC_VI Video
- AC_VO Voice

IP-Layer DSCP QoS Marking

Recognizes Cisco Fast Lane network and
sets Differentiated Services Code Point (DSCP) marking appropriately

Useful for:

- Telephony apps
- Backup and other bulk upload apps

Details to Remember

Outbound queue selection and Wi-Fi–layer QoS Marking is supported on all devices

- Only applies to outbound packets

IP-Layer DSCP QoS Marking:

- Only for outbound packets
- Only on enterprise networks with compatible Cisco equipment
- Only applies to iOS (not macOS, tvOS, watchOS)
- Only supported on Wi-Fi, not Ethernet
- Only for apps that the network administrator allows

Network Service Type Best Practices

Choose Network Service Type wisely

- Most traffic should be Best Effort
- Large transfers, not time-critical, should be Background (e.g., backup)

Network Service Type is not a priority level

Network Service Type selects

- Low throughput, low delay
- High throughput, higher delay (default)
- Scavenger traffic (only use idle capacity that otherwise would be wasted)

Summary

We're ready for Smart Queue Management and Explicit Congestion Notification

- Call to action to ISPs and mobile carriers

Support IPv6

- Both clients and servers

Support international text

- UTF-8 is the new ASCII

Express intent to networking layers

- Express when you don't want cellular
- Express when you want low throughput and low latency

More Information

https://developer.apple.com/wwdc16/714

Related Sessions

711 NSURLSession: New Features and Best Practices	Presidio	Thursday 10:00AM
706 What's New in Security	Nob Hill	Tuesday 5:00PM
201 Internationalization Best Practices	Mission	Tuesday 9:00AM
232 What's New in International User Interfaces	Nob Hill	Friday 9:00AM
710 What's New in HomeKit	Nob Hill	Wednesday 5:00PM
504 What's New in HTTP Live Streaming	Mission	Wednesday 3:00PM
234 What's New in ResearchKit	Nob Hill	Friday 10:00AM

Labs

Networking Lab 1	Frameworks Lab B	Thursday 4-6 PM
Cisco Wi-Fi Networking Lab	Fort Mason	Friday 12-2 PM
Networking Lab 2	Frameworks Lab D	Friday 2-5 PM

WWDC16