Service Discovery For IP Applications

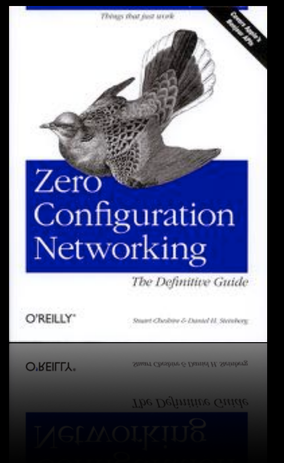
Dr Stuart Cheshire, Apple DEST

IETF 106, Singapore, Sunday 17th November 2019

About the Presenter

Dr Stuart Cheshire

- BA, Sidney Sussex College, Cambridge
- Ph.D., Stanford University, California
- Co-chairman, IETF Zeroconf Working Group
- Served term on Internet Architecture Board
- Author of O'Reilly Zero Configuration networking book
- Apple DEST (Distinguished Engineer, Scientist & Technologist)



Zeroconf Principles

AppleTalk's famous ease of use, for IP

No need to type IP addresses to…

- manually configure a device
- connect to a service

Just like people (generally) don't need to type MAC addresses

Why Do You Care?

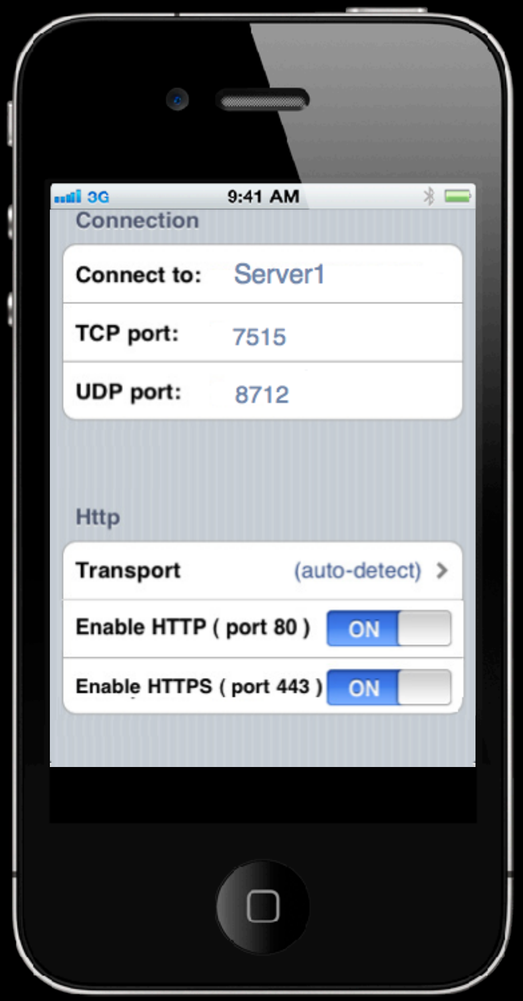
Lower support costs

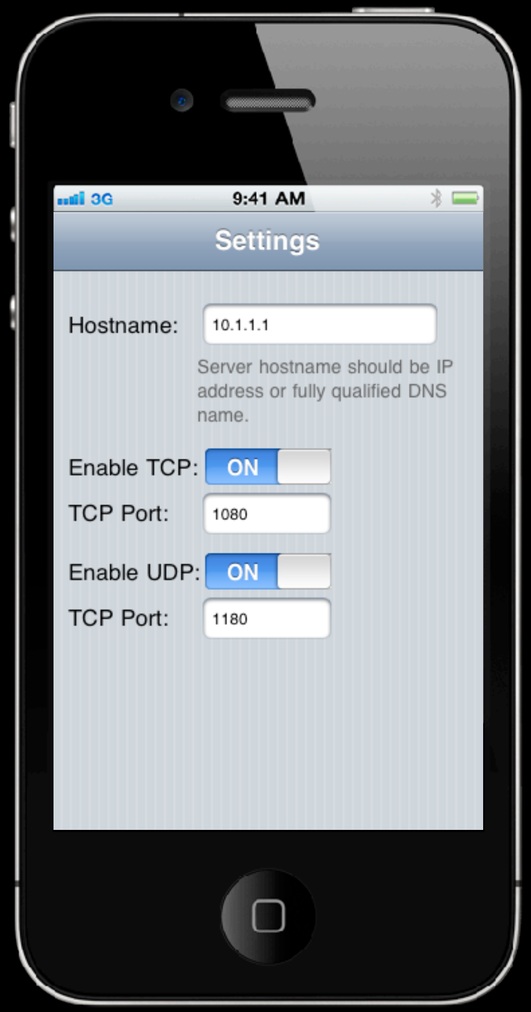
Fewer product returns

New product categories

Network products that are a joy to use

Painful Manual Configuration





No Typing IP Addresses

Apple’s Zeroconf Use Cases

AirPrint

AirPlay

HomeKit

Headless devices like the old Apple AirPort Wi-Fi access points

- No screen
- No keyboard
- No serial console
- Absolutely have to be managed over the network

Zeroconf Technologies

Addressing

Naming

Service Discovery

Addressing

DHCP for IPv4 is great (RFC 2131)

- Also want Self-Assigned IPv4 Link-Local Addresses (169.254/16)
- RFC 3927

SLAAC for IPv6 is great (RFC 4862)

- Also want Self-Assigned IPv6 Link-Local Addresses (FE80::/10)
- Also in RFC 4862

Self-Assigned Link-Local Addresses

- Pick candidate address randomly
- Check if already in use on this link; if so, try again

Naming

DNS is great (RFC 1034, RFC 1035)

When suitable DNS infrastructure unavailable, Multicast DNS can substitute

- RFC 6762
- Pick desired name, ending in ".local."
- Check if already in use on this link; if so, pick another and try again
- Ongoing conflict checking

Can type "printer.local." into a web browser, or "ssh mymac.local"

But… user needs to know what name to enter

Service Discovery Principles

Offer

- Device with listening socket publishes service on network

Enumerate

- Device seeking service discovers list of available instances

Use

- Device uses chosen service instance
- May happen once (e.g., provisioning new device on network)
- May happen repeatedly (e.g., printing to selected printer)

DNS-Based Service Discovery

We already need DNS for naming

Can we leverage that code for Service Discovery too?

Offer (Publish Service)

Service uses API to publish DNS PTR, SRV and TXT records describing its service

Enumerate (Browse)

_ipp._tcp.local. PTR

Sales._ipp._tcp.local.

Marketing._ipp._tcp.local.

Engineering._ipp._tcp.local.

3rd Floor Copy Room._ipp._tcp.local.

Use (Resolve + Connect)

3rd Floor Copy Room._ipp._tcp.local.	SRV	0	0	631	my-printer.local.
3rd Floor Copy Room._ipp._tcp.local.	TXT	pdl=application/postscript			
my-printer.local.	A	169.254.12.34			

Dynamic Ports

DNS SRV record (RFC 2782) provides port number

SRV 0 0 631 my-printer.local.

Eliminates need for a predefined well-known port

Allows multiple independent instances of a service…

- on the same host
- behind the same NAT gateway

Use (Resolve + Connect)

3rd Floor Copy Room._ipp._tcp.local.	SRV	0	0	631	my-printer.local.
3rd Floor Copy Room._ipp._tcp.local.	TXT	pdl=application/postscript			
my-printer.local.	A	169.254.12.34			

Structured Service Names

3rd Floor Copy Room . _ipp._tcp . local .

User-Visible Instance Name
Arbitrary UTF-8 Rich Text

Service Type
(Application Protocol Name)

Domain

This name is what we use to identify a service instance
No hidden GUIDs or other hidden identifiers

Structured Service Names

3rd Floor Copy Room . _ipp._tcp . local .

User-Visible Instance Name
Arbitrary UTF-8 Rich Text

Service Names & Host Names

Two Kinds of Name

Host Names

- Often used via typing on a command-line
- E.g., ssh my-computer.local.
- Restricted to US ASCII letters, digits, and hyphens

Service Instance Names

- Arbitrary UTF-8 rich text
- Entered once at setup time
- Typically selected by clicking, not typing

No MAC Addresses in Names

Printer Model Name (0001E65CD7A8)
Printer Model Name (0001E6C3E3AF)
Printer Model Name (0001E6BA565A)
Printer Model Name (0001E61945A7)
Printer Model Name (0001E6833091)

No MAC Addresses in Names

Names do not need to be made unique in the factory

- Multicast DNS has name conflict detection

Consider real end-user scenarios

Structured Service Names

3rd Floor Copy Room . _ipp._tcp . local .

Service Type
(Application Protocol Name)

Service Types (Service Names)

Unique application protocol identifier string for every different service type

- Maximum 15 characters
- US-ASCII, letters, digits and hyphens

Protocol type string

- _tcp for application protocols that run over TCP
- _udp for everything else

Service Type signifies

- *What* the service does
- *How* it does it — i.e., what on-the-wire protocol it uses

Example Service Types

_ipp._tcp	Internet Printing Protocol
_ssh._tcp	Secure Shell Remote login
_rfb._tcp	Remote Frame Buffer (VNC)
_http._tcp	Hypertext Transfer Protocol (HTML web UI over HTTP)
_daap._tcp	Digital Audio Access Protocol (Audio streaming)

Service Types

IANA manages registry of unique service type strings

RFC 6335 “IANA Procedures for the Management of the Service Name and Transport Protocol Port Number Registry”

IANA list of assigned service type strings

- http://www.iana.org/assignments/service-names-port-numbers

Applying for your own is easy (and free)

- http://www.iana.org/form/ports-services

Before shipping, register your unique service type

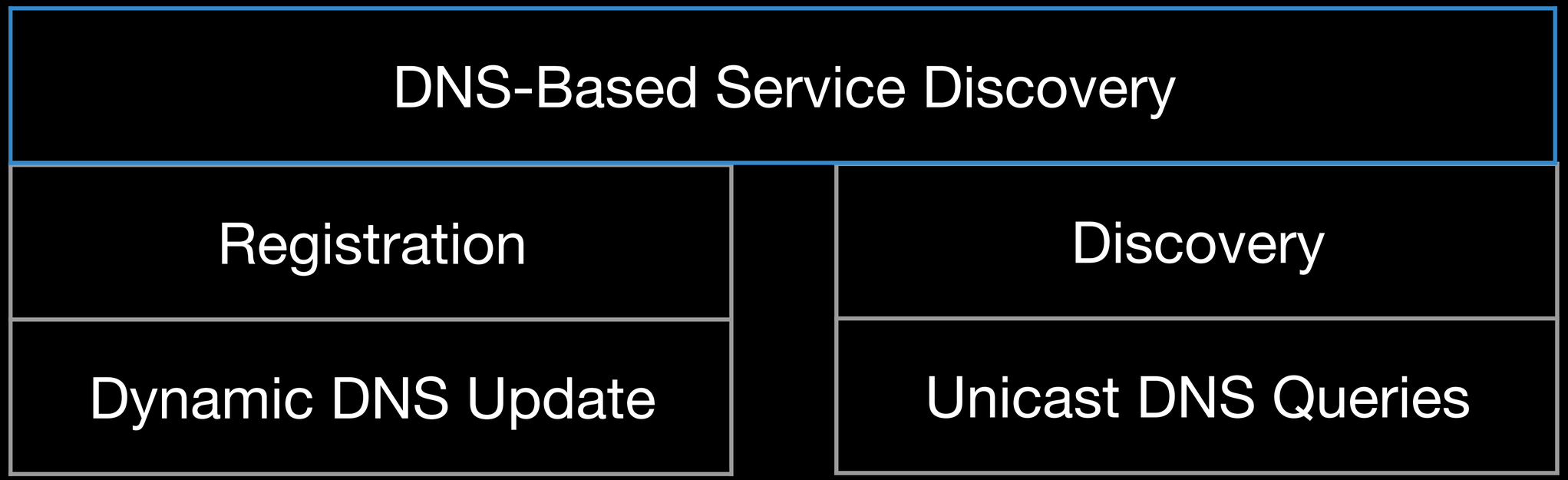
Structured Service Names

3rd Floor Copy Room . _ipp._tcp . local .

Domain

When domain is not “local”
standard unicast DNS is used instead of Multicast DNS

Wide-Area Discovery



IETF Meeting Printer Discovery

What actually happens behind the scenes when you print at IETF meetings

- When you press ⌘-P on a Mac
- When you press the AirPrint button on iOS

You can follow these steps on your own computer

IETF Meeting Printer Discovery

Info from DHCP server — Option_15 is Domain Name

```
% scutil
> list
  ...
  subKey [74] = State:/Network/Service/21B5304C...54B28F4CA1D2/DHCP
  ...

> show State:/Network/Service/21B5304C...54B28F4CA1D2/DHCP
<dictionary> {
  Option_15 : <data> 0x6d656574696e672e696574662e6f7267
  ...
}

% echo 6d656574696e672e696574662e6f7267 0A | xxd -r -p
meeting.ietf.org
```

IETF Meeting Printer Discovery

Query to check if we should perform Wide-Area Discovery

```
% dig lb._dns-sd._udp.meeting.ietf.org. ptr

; <<>> DiG 9.6-ESV-R4-P3 <<>> lb._dns-sd._udp.meeting.ietf.org. ptr
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35624
;; flags: qr aa rd ra;
                    QUERY: 1, ANSWER: 1, AUTHORITY: 2, ADDITIONAL: 4

;; QUESTION SECTION:
;lb._dns-sd._udp.meeting.ietf.org. IN PTR

;; ANSWER SECTION:
lb._dns-sd._udp.meeting.ietf.org. 3600 IN PTR meeting.ietf.org.
```

IETF Meeting Printer Discovery

Querying 8.8.8.8 gets the same answer

```
% dig @8.8.8.8 lb._dns-sd._udp.meeting.ietf.org. ptr

; <<>> DiG 9.6-ESV-R4-P3 <<>> @8.8.8.8 lb._dns-sd._udp.meeting.ietf.org. ptr
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24571
;; flags: qr rd ra; QUERY:1, ANSWER:1, AUTHORITY:0, ADDITIONAL:0

;; QUESTION SECTION:
;lb._dns-sd._udp.meeting.ietf.org. IN PTR

;; ANSWER SECTION:
lb._dns-sd._udp.meeting.ietf.org. 1532 IN PTR meeting.ietf.org.
```

IETF Meeting Printer Discovery

Finding available printers on macOS

```
% dig +short _pdl-datastream._tcp.meeting.ietf.org. ptr
term-printer._pdl-datastream._tcp.meeting.ietf.org.
```

IETF Meeting Printer Discovery

Printing on macOS

```
% dig +short term-printer._pdl-datastream._tcp.meeting.ietf.org. srv
0 0 9100 term-printer.meeting.ietf.org.

% dig +short term-printer.meeting.ietf.org. AAAA
2001:df8::48:200:74ff:fee0:6cf8
```

IETF Meeting Printer Discovery

Finding available printers on iOS

```
% dig +short _universal._sub._ipp._tcp.meeting.ietf.org. ptr
term-printer._ipp._tcp.meeting.ietf.org.
```

IETF Meeting Printer Discovery

Printing on iOS

```
% dig +short term-printer._ipp._tcp.meeting.ietf.org. srv
0 0 631 term-printer.meeting.ietf.org.

% dig +short term-printer.meeting.ietf.org. aaaa
2001:df8::48:200:74ff:fee0:6cf8
```

DNS-SD Subtypes

Finding available printers on iOS

```
% dig +short _universal._sub._ipp._tcp.meeting.ietf.org. ptr
term-printer._ipp._tcp.meeting.ietf.org.
```

The _universal subtype indicates that the iPhone is looking only for IPP printers that support driverless printing using Universal Raster Formal

Subtypes allow simple filtering to limit results to a subset

Implementations

Apple's Apache 2 Open Source mDNSResponder

- https://github.com/IETF-Hackathon/mDNSResponder

Avahi LGPL Open Source (GNU Lesser General Public License v2.1)

- https://www.avahi.org/

Included in macOS, iOS

Included in most Linux distributions

Included in Android "Jelly Bean" (API Level 16, June 2012) and later

Included in Windows 10 (July 2015) and later

Programming with DNS-SD

Evaluation using dns-sd command-line tool

Offer (register) service	`dns-sd -R <Name> <Type> <Domain> <Port>`
Enumerate (browse)	`dns-sd -B <Type> <Domain>`
Use (lookup/resolve)	`dns-sd -L <Name> <Type> <Domain>`

Selecting Local or Wide-Area

API is the same for both

Set domain to “local” for local publishing and discovery

Set domain to something else for wide area publishing and discovery

Selecting Local or Wide-Area

Recommended:

For Register and Browse, **set domain to empty string or NULL,**
to let API automatically respect system configuration

For Resolve, set domain to the domain value you discovered
in the Browse results

This is what allows printer discovery to work at IETF meetings

Programming with DNS-SD

Evaluation using dns-sd command-line tool

```
dns-sd -R Test _test._tcp "" 123

dns-sd -B      _test._tcp

dns-sd -L Test _test._tcp
```

Programming with C APIs

C APIs defined in dns_sd.h, available on:

- Apple's macOS and iOS
- Linux, with mDNSResponder or Avahi
- Windows, with Apple's Bonjour for Windows installed
  - Easiest way to get Bonjour for Windows:
    Install Bonjour Print Services for Windows
    https://support.apple.com/downloads/bonjour_for_windows

Programming on Android

Introduced in Android “Jelly Bean” (API Level 16, June 2012)

https://developer.android.com/reference/android/net/nsd/NsdManager.html

https://developer.android.com/training/connect-devices-wirelessly/nsd

Programming on Windows 10

Introduced in Windows 10 (July 2015)

https://channel9.msdn.com/Events/Build/2015/3-79

https://docs.microsoft.com/en-us/uwp/api/windows.networking.servicediscovery.dnssd

Programming with C APIs

```
DNSServiceRegister
(
    DNSServiceRef               *sdRef,
    DNSServiceFlags              flags,
    uint32_t                     interfaceIndex,
    char                        *name,
    char                        *regtype,
    char                        *domain,
    const char                  *host
    Opaque16                     port,
    uint16_t                     txtLen
    char                        *txtRecord,
    DNSServiceRegReply           callBack,
    void                        *context
);
```

Programming with C APIs

```
DNSServiceBrowse
(
    DNSServiceRef               *sdRef,
    DNSServiceFlags              flags,
    uint32_t                     interfaceIndex,

    char                        *regtype,
    char                        *domain,




    DNSServiceBrowserReply       callBack,
    void                        *context
);
```

Programming with C APIs

```
DNSServiceResolve
(
    DNSServiceRef                *sdRef,
    DNSServiceFlag                flags,
    uint32_t                      interfaceIndex,
    char                         *name,
    char                         *regtype,
    char                         *domain,




    DNSServiceResolverReply       callBack,
    void                         *context
);
```

Programming with C APIs

All calls are asynchronous

- Note that all take a callback function pointer
- All return a DNSServiceRef encapsulating the asynchronous operation

Call DNSServiceRefSockFD(sdRef) to get underlying file descriptor

Add to your existing event loop (select, poll, kevent, etc.)

When event happens, call DNSServiceProcessResult(sdRef)

Your supplied callback function will be invoked

See dns-sd.c source for sample code

Asynchronous Results

Don't expect immediate results from DNSServiceBrowse

- Usually the network should be fast, but sometimes it might not be

Use live UI that continues to show results as they arrive

Note that no Apple network browsing UI (AirDrop, AirPlay, AirPrint, etc.) has a "refresh" button

Event Notification

Continuous asynchronous results means that DNS-SD also provides event notification, using the same APIs for both local and remote

Server

- Publish service: `DNSServiceRegister`
- Update TXT record keys: `DNSServiceUpdateRecord`

Client

- Monitor for changes: `DNSServiceQueryRecord`

Event Notification

Local Event Notification

- Publisher announces changes via Multicast DNS
- Subscriber receives multicast announcement on local link

Remote Event Notification

- Publisher updates service registry
- Registry notifies interested clients
- draft-ietf-dnssd-push (DNS Push Notifications)
- RFC 8490 DNS Stateful Operations

DNSServiceBrowse — The Right Way

No refresh button

No open-ended browsing

Browse when requested by user, not constantly

Stop browsing when not displaying browse UI

UI design: Use windows, not pull-down menus

- Traditionally, menus not expected to change once displayed
- Users generally more comfortable with window content updating

Resolve and connect when requested by user, not every service you find

Storing Results

Bad ideas

- Save just the IP address
- Save the IP address and port number
- Save the host name and port number

The right way

- Late binding is the key
- Service is identified by three-tuple: { Name, Type, Domain }
- Save { Name, Type, Domain } tuple
- Resolve on demand at time of use

Don’t Resolve Everything!

Joe’s Printer	joe.local	9100	pdl=application/postscript …
Sally’s Printer	sally.local	9100	pdl=application/postscript …
Jim’s Printer	jim.local	9100	pdl=application/postscript …
Penny’s Printer	penny.local	9100	pdl=application/postscript …
Paul’s Printer	paul.local	9100	pdl=application/postscript …
Mary’s Printer	mary.local	9100	pdl=application/postscript …

Resolve on Demand

Don’t DNSServiceResolve until you need to use service

Resolving causes extra network traffic

IP address might be out of date by the time you use it

Always save and work with the service name

Only DNSServiceResolve when ready to use service

Happy Eyeballs

DNSServiceResolve may return multiple possible IP addresses

- For best user experience, try all of them, with staggered start times
- RFC 8305 Happy Eyeballs Version 2

Can implement this yourself, or use an API like Apple's Network.framework that does Happy Eyeballs Version 2 for you:

```
let conn = NWConnection(to:
    .service(name: "Test", type: "_test._tcp.", domain: "local.", interface: nil),
    using: .tcp)
```

Benefits of DNS-SD

Handling low reliability of Wi-Fi multicast

- Query retransmission
- Exponential backoff

Handling low speed of Wi-Fi multicast

- Duplicate query suppression
- Systemwide opportunistic caching
- Long cache lifetimes
- Known-answer lists suppress unnecessary answers
- Duplicate answer suppression

Benefits of DNS-SD

Fast discovery

- Service announcements
- Exponential backoff

Name management

- Name conflict detection
- Automatic renaming
- Ongoing monitoring

Benefits of DNS-SD

Fast removal

- Goodbye packets on shutdown
- Automatic reconfirm upon connection failure
- Passive Observation of Failures (POOF)

Mobility

- Sleep/wake handling
- Network connect/disconnect
- Wi-Fi access point roaming

Benefits of DNS-SD

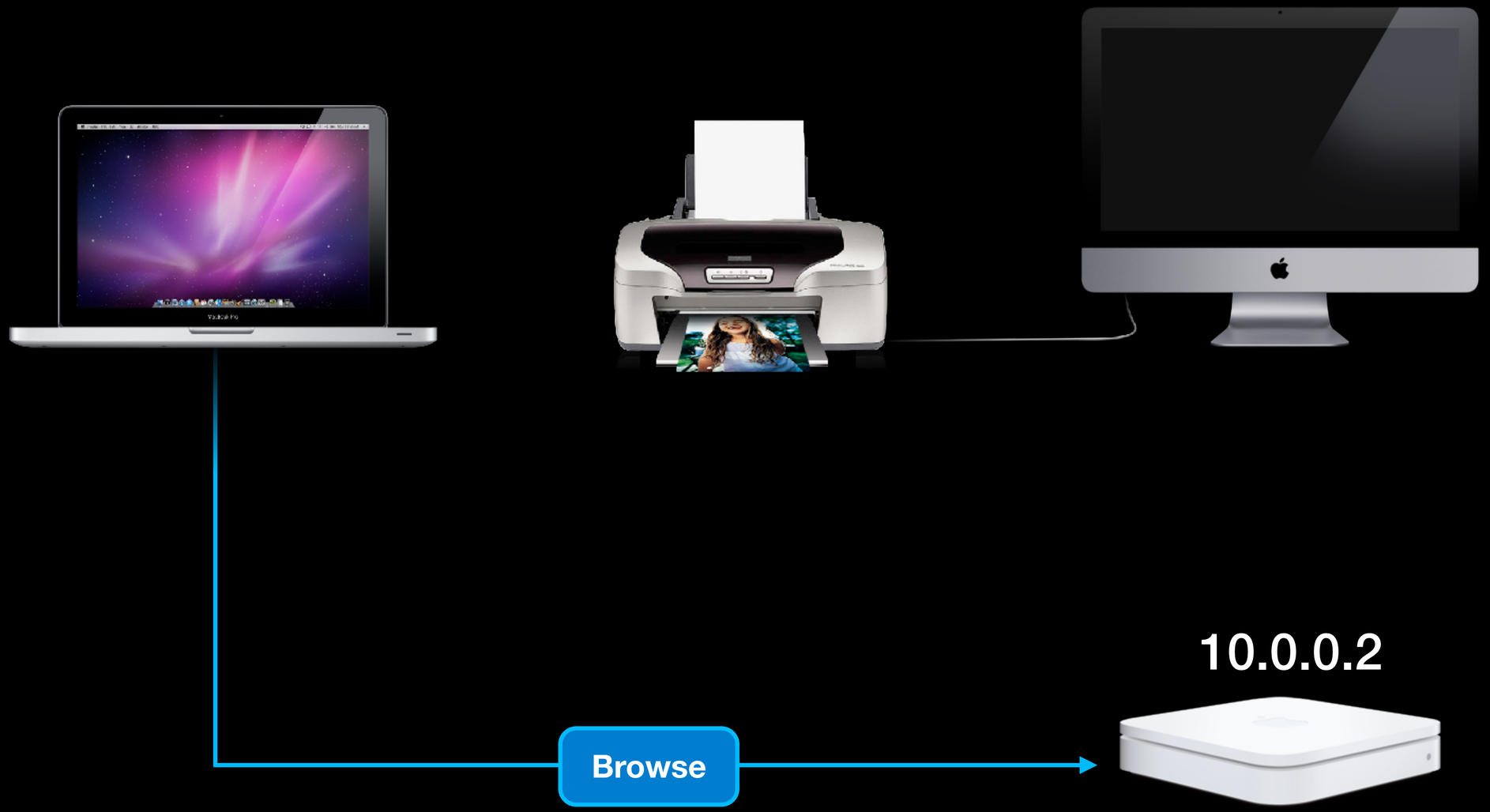
Sleepy devices

- Hand off records to Sleep Proxy
- Either network sleep proxy, or internal sleep proxy

Remote discovery via unicast

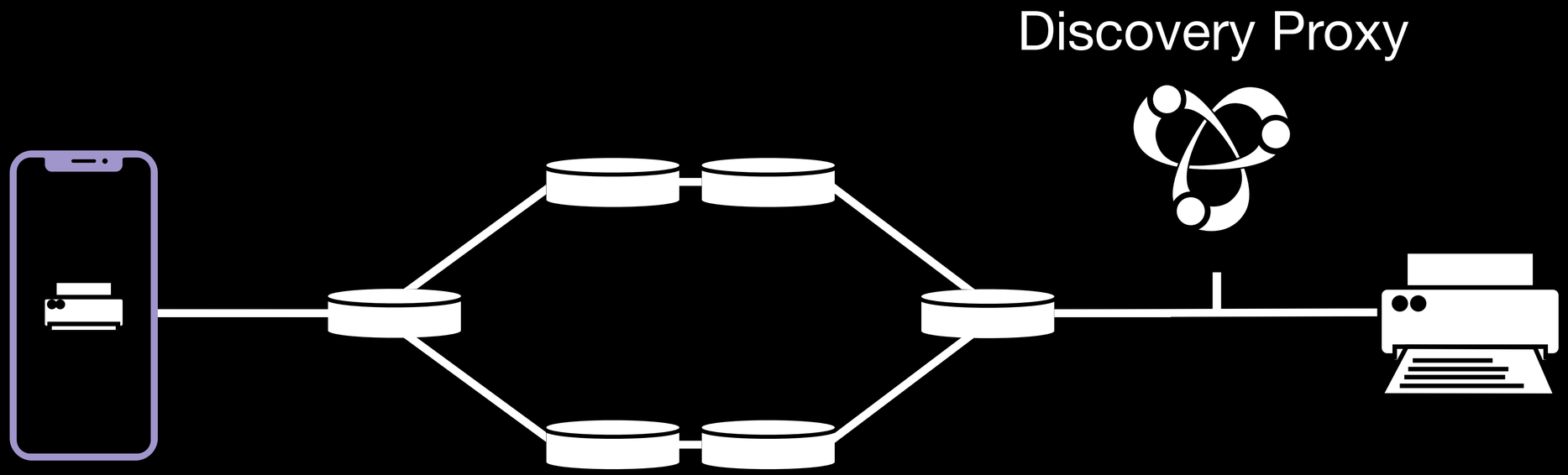
- Manually entered data (like IETF Terminal Room printer)
- Automatic via DNS Update & Service Registration Protocol
- Automatic via Discovery Proxy
- Live updates via DNS Push Notifications

Sleep Proxy — Enumerate



Sleep Proxy — Use

Discovery Proxy



Resources

Addressing RFCs

- RFC 3927 Dynamic Configuration of IPv4 Link-Local Addresses
- RFC 4862 IPv6 Stateless Address Autoconfiguration

Resources

Base Service Discovery RFCs

- RFC 6335 IANA Procedures for Service Name Registration
- RFC 6760 Requirements to Replace AppleTalk Name Binding Protocol
- RFC 6761 Special-Use Domain Names
- RFC 6762 Multicast DNS
- RFC 6763 DNS-Based Service Discovery

Resources

Extension Service Discovery RFCs

- RFC 3007 Secure Domain Name System (DNS) Dynamic Update
- draft-sctl-service-registration (Service Registration Protocol)
- RFC 8490 DNS Stateful Operations
- draft-ietf-dnssd-push (DNS Push Notifications)
- draft-ietf-dnssd-hybrid (Discovery Proxy)
- draft-sctl-dnssd-mdns-relay (Multicast DNS Discovery Relay)
- draft-cheshire-dnssd-roadmap (Service Discovery Road Map)

Resources

Debugging

- Source code: https://github.com/IETF-Hackathon/mDNSResponder
- dns-sd command-line tool
- Mac App: Discovery - DNS-SD Browser
- Apple’s Bonjour Conformance Test https://developer.apple.com/softwarelicensing/bonjour/

Call to Action

Software Developers

- Use DNS-SD to advertise and discover network services

Hardware Developers

- Build all three legs of Zeroconf into your hardware products
- Don’t skip Link-Local Addressing — both IPv4 and IPv6
- Use Apple Bonjour conformance test to verify

Q&A

Please help IETF EDU team
by completing their short five-question survey

https://www.surveymonkey.com/r/106service