SCIENCES SORBONNE UNIVERSITÉ

The Bonjour protocol

Fabrice.Kordon@lip6.fr

LIP6

interoperability without configuration

Zero configuration networking

Peer-to-peer thanks to bonjour

- Implementation of the ZeroConf standard
- RFC 3927 (IETF, May 2005)

High-level network service offered by the OS

- Publication of a service
- Browse of available services

Available on a local network

Publish a service

One service = one object (NS)NetService

- To be specified : name, domain, type and port
- Delegation handles asynchronous informations
- How to do «à la Swift»

```
let netService = NetService(domain: "local",
                            type: "_foo._tcp.",
                            name: "I like Apples",
                            port: 9090)
netService.delegate = self
netService.publish()
```

  - **Similar in Objective-C**

- Attribute includesPeerToPeer
  - **Activate bluetooth**
- Once published, use delegation to manage answers
  - **(NS)NetServicesDelegate**

Publish a service

One service = one object (NS)NetService

- To be specified : name, domain, type and port
- Delegation handles asynchronous informations
- How to do «à la Swift»

```
let netService = NetService(domain: "local",
                            type: "_foo._tcp.",
                            name: "I like Apples",
                            port: 9090)
netService.delegate = self
netService.publish()
```

  - **Similar in Objective-C**

- Attribute includesPeerToPeer
  - **Activate bluetooth**
- Once published, use delegation to manage answers
  - **(NS)NetServicesDelegate**

NetServiceDelegate

Optional methods

- Handling publication

```
func netServiceWillPublish(_ sender: NetService)

func netService(_ sender: NetService,
                didNotPublish errorDict: [String : NSNumber])

func netServiceDidPublish(_ sender: NetService)
```

- Handling service resolution

```
func netServiceWillResolve(_ sender: NetService)

func netService(_ sender: NetService,
                didNotResolve errorDict: [String : NSNumber])

func netServiceDidResolveAddress(_ sender: NetService)
```

- Data update & termination

```
func netService(_ sender: NetService,
                didUpdateTXTRecord data: Data)

func netServiceDidStop(_ sender: NetService)
```

NetServiceDelegate

Optional methods

- Handling publication

```
func netServiceWillPublish(_ sender: NetService)

func netService(_ sender: NetService,
                didNotPublish errorDict: [String : NSNumber])

func netServiceDidPublish(_ sender: NetService)
```

- Handling service resolution

```
func netServiceWillResolve(_ sender: NetService)

func netService(_ sender: NetService,
                didNotResolve errorDict: [String : NSNumbe

func netServiceDidResolveAddress(_ sender: NetService)
```

- Data update & termination

```
func netService(_ sender: NetService,
                didUpdateTXTRecord data: Data)

func netServiceDidStop(_ sender: NetService)
```

NetServiceBrowser

Looking for published services

Principles

- Creation
- Assignment of the delegate
- Start research

```
let myBrowser = NetServiceBrowser()
myBrowser.delegate = self
myBrowser.searchForServices(ofType: "_foo._tcp.",
                            inDomain: "local")
```

Other methods…

- stop()
- etc.

NetServiceBrowserDelegate

Optional methods

- Handling domains

```
func netServiceBrowser(_ browser: NetServiceBrowser,
                       didFindDomain domainString: String,
                       moreComing: Bool)

func netServiceBrowser(_ browser: NetServiceBrowser,
                       didRemoveDomain domainString: String,
                       moreComing: Bool)
```

- Handling services

```
func netServiceBrowser(_ browser: NetServiceBrowser,
                       didFind service: NetService,
                       moreComing: Bool)

func netServiceBrowser(_ browser: NetServiceBrowser,
                       didRemove service: NetService,
                       moreComing: Bool)
```

- Others

```
func netServiceBrowserWillSearch(_ browser: NetServiceBrowser)

func netServiceBrowser(_ browser: NetServiceBrowser,
                       didNotSearch errorDict: [String : NSNumber])

func netServiceBrowserDidStopSearch(_ browser: NetServiceBrowser)
```

Information exchange

By means of a shared dictionary

- Store additional informations
  - **In a NetService object (get/set/conversion)**

```
func txtRecordData() -> Data?
func setTXTRecord(_ recordData: Data?) -> Bool

class func data(fromTXTRecord txtDictionary: [String : Data]) -> Data
class func dictionary(fromTXTRecord txtData: Data) -> [String : Data]
```

- May be watched by peers
  - **Notification when changes are published**
- Handled by NSNetServiceDelegate

```
func netService(_ sender: NetService,
                didUpdateTXTRecord data: Data)
```

- Stopping monitoring such data
  - **In the didRemoveService**
  - **Method stopMonitoring()**

Information exchange

By means of a shared dictionary

Conversions String/data

String to Data (String method)

```
func data(using encoding: UInt) -> Data?
```

Data to String (Data method)

```
func base64EncodedString(options: NSData.Base64EncodingOptions = []) -> String
```

- May be watched by peers
  - **Notification when changes are published**
- Handled by NSNetServiceDelegate

```
func netService(_ sender: NetService,
                didUpdateTXTRecord data: Data)
```

- Stopping monitoring such data
  - **In the didRemoveService**
  - **Method stopMonitoring()**

Handling shared information

Importance of conventions

- Inside an App
- Between Apps (when required)

Process

- Fetch the data (by delegation)
- Transform the data into a dictionary

```
class func dictionary(fromTXTRecord txtData: Data) -> [String : Data]
```

- Fetch informations from the dictionary
  - **Conventions = keys**
  - **Convention = cast of associated Data into the desired type**

As a conclusion…

Useful mechanism

- Allow to exchange simple informations
- Flexible and dynamic
- Easy configuration for users (no configuration)

you MUST be ready to use it

- Even if limited to local exchanges

Also works in Android

- NetWork Service Discovery
  - **Introduced in «Jelly bean» (2012)**