A Java API for unifying ad-hoc Wifi networking

Peter Banis, Klaus Cipi, Michael Kolar, Robert Olsen

Faculty Sponsor: Dr. Marius Silaghi

Milestone 2 (October 29)

- Divorce the existing source code from DD (Direct Democracy) app
- Implement OS detection
- Create new Exceptions (Integrity check)
- Implement logging for the API
- Created configuration class for each OS
- Integration and Unit testing

Milestone 2 Progress (1/2)

Task	Completion %	Peter	Klaus	Michael	Robert	To Do
Divorce the existing source code from DD (Direct Democracy) app	100%	25%	25%	25%	25%	None
Automatic OS detection (IT&D)	100%	20%	50%	15%	15%	None
Create new Exceptions (Integrity check)	100%	40%	30%	15%	15%	None

Milestone 2 Progress (2/2)

Task	Completion %	Peter	Klaus	Michael	Robert	To Do
Implement logging for the API	100%	50%	20%	15%	15%	None
Created configuration class for each OS	100%	25% (Linux)	25% (Mac)	25% (Windows)	25% (Windows)	None
Integration and Unit testing	100%	25%	25%	25%	25%	None

System Configuration (1/2)

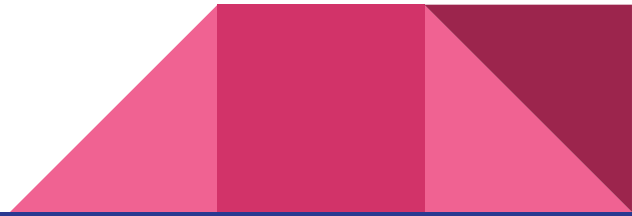
- Each OS has a unique way of creating and connecting to ad-hoc networks
- Abstract Java class SystemConfiguration with concrete subclasses for Windows, Mac and Linux
- Each concrete class handles the OS specific operations without exposing developers to this problem
- Key functions are createNetwork() and connectToNetwork()

System Configurations (2/2)

```
import ...

public abstract class SystemConfiguration {
    private int channel = 11;
    private String SSID;
    private String password;
    private String networkInterface;

    public int getChannel() { return channel; }

    public String getPassword() { return password; }

    public String getSSID() { return SSID; }

    public String getNetworkInterface() { return networkInterface; }

    public abstract String[] getInterfaces() throws ScriptFailureException, InterruptedException;

    public abstract String getCurrentInterface() throws ScriptFailureException, IOException, InterruptedException;

    public abstract int[] getSupportedChannels(String wirelessInterface) throws ScriptFailureException;

    public abstract int connectToNetwork() throws ScriptFailureException ;

    public abstract int connectToNetwork(String networkName, String password, String interfaceName, int channel) throws ScriptFailureException ;

    public abstract int createNetwork() throws ScriptFailureException;

    public abstract int createNetwork(String networkName, String password, String interfaceName, int channel) throws ScriptFailureException;

    public abstract int disconnectFromNetwork() throws ScriptFailureException;
```

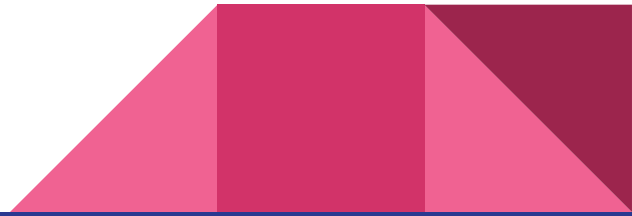
Windows (1/2)

- Scripts were made which would allow various functions: create network, connect to network, disconnect from network, show interfaces, show current interface, and show supported channels
- SystemConfigurationWindows.java was created that implemented methods which utilized these scripts
  - It runs the scripts when needed
  - Captures any output from stdout and stderr
  - Handles any errors and provides useful user feedback



Windows (2/2)

- Windows could connect to Windows and Linux networks
- Mac and Windows could connect to a Windows network
- Linux couldn’t connect to the Windows network because Linux needs to be configured for ad-hoc vs infrastructure; however, Windows 10 doesn’t allow its hostednetwork to be ad-hoc
- Windows 10 removed ad-hoc network support, so we are doing research to get around this



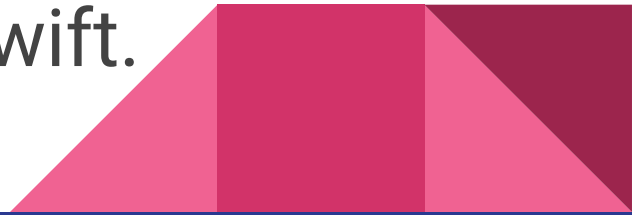
Object Factory (OS detection)

- Users only need to create a general purpose SystemConfiguration object. They declare the object and use a method called init located in the Factory class (ex: SystemConfiguration example = SystemConfigurationFactory.init())
- The init method detects which os the user has by using the System.getProperty(“os.name”) java method
- Afterwards, it creates a new instance of the relevant SystemConfiguration subclass based on the user’s os, and then returns that instance

Mac

- Problem: Some of the script were outdated and could not automate all the tasks that our API has to perform.

- Solution: Write new scripts in Swift.



Connect/Create an ad-hoc network

connect_to_network.swift

connect_to_network.swift 〉 No Selection

```swift
#!/usr/bin/swift

import Foundation
import CoreWLAN

let interface = CommandLine.arguments[1]
let ssid = CommandLine.arguments[2]
let password = CommandLine.arguments[3]

if let iface = CWWiFiClient.shared().interface(withName: interface) {
    do {
        if let avaiableNetworks = try iface.scanForNetworks(withName: ssid).first {
            try iface.associate(to: avaiableNetworks, password: ssid)
            print("Connected")
        } else {
            print("Error: network unable to connect")
            exit(1)
        }
    } catch {
        print("Error: network unable to connect")
    }
} else {
    print("Error: interface does not exists")
}

```

create_network.swift

create_network.swift 〉 No Selection

```swift
#!/usr/bin/swift

import Foundation
import CoreWLAN

let interfaceName = CommandLine.arguments[1]
let networkName = CommandLine.arguments[2]
let password = CommandLine.arguments[3]
let channels = Int(CommandLine.arguments[4])

if let iface = CWWiFiClient.shared().interface(withName: interfaceName) {
    do {
        if password == "" {
            try iface.startIBSSMode(
                withSSID: networkName.data(using: String.Encoding.utf8)!,
                security: CWIBSSModeSecurity.none,
                channel: channels!,
                password: password as String
            )
        } else {
            try iface.startIBSSMode(
                withSSID: networkName.data(using: String.Encoding.utf8)!,
                security: CWIBSSModeSecurity.WEP104,
                channel: channels!,
                password: password as String
            )
        }
    } catch {
        print("Error: network unable to start")
    }
} else {
    print("Error: interface does not exists")
}

```

Other scripts

- Disconnect from an ad-hoc network
- List all available interfaces
- Return current Wi-Fi interface
- List all the channels available for a specific interface

Wrapper class

- SystemConfigurationMac.java is the wrapper class to drive the scripts by
  - Checking the correct execution of the scripts
  - Capturing the output
  - Capturing exceptions
  - Granting permissions

Exceptions

- Inability to recognize the platform the API is on (UnknownOSException)
- Files may not be found (ScriptMissingException)
- Lack the sufficient permissions for reading/writing/executing (DeniedPermissionException)
- Failure to run scripts to completion (ScriptFailureException)
- Not enough arguments given (MissingArgumentsException)

Linux

- 5 bash scripts: Detecting wireless interfaces, detecting currently in use wireless interface, detecting supported channels for a given interface, connecting/creating a network and disconnecting from a network
- Because creating a network requires directly configuring the network interface, the Java file must be run with sudo. Various solutions were attempted without success. Considered to be an annoyance only at the moment.
- Script for making/connecting to a network has to allow for choosing network SSID, password(optional), channel and network interface.

Logging

- Use Java.util.logging provides a class to handle logging information in the API
- Each script for returns information. For example, if the script cannot be found we receive “exit code 2: cannot find file or directory”
- This information is logged in a static java object
- Upon API exit a log file is written containing all the logged information

Sample Program

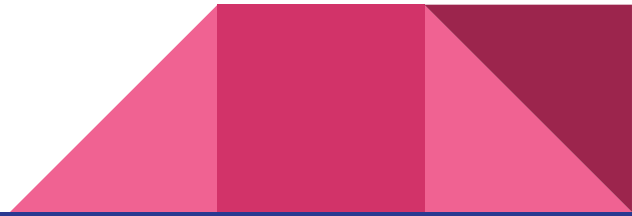
```
1  public class Test {
2
3      public static void main(String[] args) {
4        try {
5            // Creating an object specific OS from the SCFactory
6            SystemConfiguration testing = SystemConfigurationFactory.init();
7            // Test getCurrentInterfaces
8            System.out.println(testing.getInterfaces());
9            //Test createNetwork: SSID "admin"
10           testing.createNetwork("admin", "pass", testing.getCurrentInterface(), 11);
11           //Test stopHostNetwork
12           testing.disconnectFromNetwork();
13
14       } catch(Exception e) {
15         e.printStackTrace();
16       }
17     }
18 }
19
```

Connection Capabilities

OS connects to...	Windows Network	Linux Network	Mac OS Network
Windows	Yes	Yes	No
Linux	No	Yes	Yes
Mac OS	Yes	Yes	Yes

Milestone 3 (November 26)

- Divorce API Networking code from Direct Democracy application
- Expand and create networking functions
- Acquire DirectP2P capable adapters and Android phones for future testing
- Implement support for configuring devices in DirectP2P mode
- Create more specific exceptions
- Debug and correct connection problems between Windows 7/10 and Mac/Linux



Milestone 3 Matrix

Task	Peter	Klaus	Michael	Robert
Divorce API Networking code from Direct Democracy application	25%	25%	25%	25%
Expand and create networking functions	25%	25%	25%	25%
Acquire DirectP2P capable adapters and Android phones for future testing	25%	25%	25%	25%
Implement support for configuring devices in DirectP2P mode	25%	25%	25%	25%
Create more specific exceptions	25%	25%	25%	25%
Debug and correct connection problems between Windows 7/10 and Mac/Linux	25%	25%	25%	25%

Questions?