- ODTONE -

- **O**pen **D**ot **T**wenty **One** -

Installation Manual & User Guide

Simao Reis, Rui Costa

February 12, 2010

Contents

1 ODTONE Installation

As ODTONE is developed in order to work in several platforms, we therefore won't provide detailed tutorials for each mainstream operating system. We will rather present the main guidelines for installing ODTONE and any relevant notes on specific platforms that might prove an obstacle.

1.1 Required Dependencies

In order to correctly use ODTONE you must first make sure you have the required dependencies installed.

- *Boost* Developer libraries (v1.37 or higher)

  If you do not already have them in your system, you can go to `http://www.boost.org/` and download a more recent version of the *Boost* libraries and manually compile them.

- *SQLite* Developer library (v3.5.9 or higher)

  If you do not already have them in your system, you can go to `http://www.sqlite.org/` and download a more recent version.

1.2 Getting ODTONE

- In a *tarball*

  You can get the ODTONE source code *tarball* from our project page, at `http://helios.av.it.pt/projects/odtone/files`. Then just unpack it and you are ready to compile and install ODTONE.

- From the *git* repository You can *clone* the *git* repository, from http://helios.av.it.pt/projects/odtor
  therefore creating your own local copy.

1.3 Compilling and installing ODTONE

Once you have obtained the ODTONE source code and have assured you have the right dependencies are installed in all machines you intend to use ODTONE, then you are ready to compile and install.

- Windows

  Using *Visual Studio 2008*, choose the option to open an existing project. Browse to the folder *vc90/* on your ODTONE folder and select and open the ODTONE project. Then just build the project.

NOTE that this is still an *alpha* version, therefore in future releases *Windows* support will be improved, in this manual.

- Linux/Unix

  You just need to run the "*./autogen.sh*" script and then run `./configure` and finally `make` to compile the code. Install is not yet required, and therefore all applications are local and can be found under the *src/* folder.

2 `ODTONE` User Guide

For now the `MIHF`s capabilities are read from configuration files. Future versions will inquire the available `Link SAP`s for the required information.

2.1 Local Demo

The local demo consists in a simple experiment to demonstrate simple message exchange between a `MIHF`, a `MIH_User` and a `Link_SAP`. It allows you to see how evenst are generated by the `Link_SAP` and reported to a `MIH_User` that has subscribed to these events.

2.1.1 Configuration

To run the local demo you will need to edit the default configuration file *odtone.conf* that is stored in the *src/* directory. The file is well documented and you will need to edit the *link_ addr_ list* entrance and add the MAC address of your computer's network cards.

2.1.2 Running the Demo

After editing the configuration file, to run this `ODTONE` demo the best way is to open 3 terminals. On one terminal start `ODTONE`, and on the next terminal start the `MIH_User`.

If all went well the `MIH_User` has requested an `Capability_Discover` to the local `MIHF` and printed out some information of the interfaces you previously configured.

Now it's time to start the `Link_SAP`, on your third terminal, and send some Link indications to the `MIHF` and check that the `MIH_User` received notifications .

So that the `Link_SAP` detects some events you should now proceed to disconnect, disable or shutdown you network cable/interface or your wireless card according to your configuration file (*odtone.conf*).

2.2 Remote Demo

The remote demo consists in a local `MIH_User` obtaining notifications of events that are happening on a `Link_SAP` remotely located on another machine. You are required to have two machines for this experiment, the first machine hosting an `ODTONE` instance (`mihf1`) and the `MIH_User`, while the second will host a second `ODTONE` instance `mihf2`) and the `Link_SAP`.

2.2.1 Configuration

As was said earlier, automatic remote peer `MIHF` discovery is not yet implemented, so you need to add the remote `MIHF`s IP address and port number to the configuration file (*odtone.conf*).

Edit the file and add an entry to *peers* in the form:

```
<mihf_id> <ip> <port>
```

`<mihf_id>` is the identifier of the remote `MIHF`, `<ip>` and `<port>` are self explanatory.

NOTE that you need to edit the configuration file on both machines.

2.2.2 `mihf1` configuration

On the machine with the `mihf1` the config file can look like:

```
    [mihf]
    id = mihf1
    local_port = 1025
    remote_port = 4551
    peers = mihf2 <mihf2_IP_address> 4551
    users = user 1234
```

2.2.3 `mihf2` configuration

The configuration file of `mihf2` would be:

```
[mihf]
id = mihf2
local_port = 1025
remote_port = 4551
peers = mihf1 <mihf1_IP_address> 4551
links = link 1235
link_addr_list = 802_11 <mac_address>,ethernet <mac_address>
event_list = link_detected, link_up, link_down,
link_parameters_report, link_going_down,
link_handover_imminent, link_handover_complete
```

NOTE that you need to setup the MAC addresses of the `mihf2` interfaces.

2.2.4 Running the Demo

- On the machine hosting `mihf1`, start a terminal and run `ODTONE`. Then, on another terminal start the `MIH_User` adding the parameter `--dest mihf2`. The *--dest* option tells the `MIH_User` to set the `802.21` destination field of the frame to `mihf2`.
- On the machine hosting `mihf2`, start a terminal and run `ODTONE`. Then, on another terminal start the `Link_SAP`. So that the `Link_SAP` detects some events you should now proceed to disconnect, disable or shutdown your network cable/interface or you wireless card according to your configuration file (*odtone.conf*).