RUNES Final Demo, IPv6 and RUNES

Socrates Varakliotis, Manish Lad, Peter Kirstein

19 June 2007, Version 7

1. Introduction

The aim of this document is to outline what is being implemented concerning IPv6 and Network Mobility (NEMO) in RUNES, and what would be available to take over to the U-2010 project. In order to do this, we need to present the storyline for parts of the RUNES Final Review Demonstrator (RFD) to make this a self-standing document, and then show the IPv6 and 6LoWPAN activities that have been implemented to match this storyline. These parts of the RFD will be called the RF6D here. Then we describe the equipment at UCL for this activity, all of which could be re-used in U-2010, and its relevant software. Finally we discuss how this links into the sub-projects of U-2010 under WP4.

2. The IPv6 Demonstration Storyline

2.1 A Schematic to Illustrate the Storyline

The whole storyline of demonstrations in RUNES ties back to the CD-ROM on the fire-in-the-tunnel [1]. There is a substantial demonstration to illustrate parts of this scenario; this is the RUNES Final Demonstrator (RFD). However, in order to be self-explanatory, we will describe here only that part of the scenario which has been implemented in the IPv6 activity.

Here there is a Tunnel, which has both IPv4-enabled and IPv6-enabled wireless sensors through a sensor network; these sensors measure temperature, light and humidity and can be either:

a. fixed into the tunnel infrastructure, or

b. fixed onto the fire fighters' uniform/equipment, and/or,

c. deployed by the fire fighters in the tunnel upon arrival.

The sensors are controlled from a tunnel gateway. Through a LAN, the tunnel gateway communicates with a local control room. Via this control room, it is possible to communicate back to other devices and organisations through the Internet.

The gateway belongs to the tunnel infrastructure, and regularly relays measurements from the sensors over the 802.11.4 radio. But, fire fighters may also bring their own mobile gateway and may want to deploy new sensors to the tunnel. The fire fighters' gateway would then rely on the tunnel's 802.11 WiFi Access Points to connect to the rest of the wired network and transfer data to the control rooms (local control centre for the fire fighters or remote control room).

Figure 1 illustrates the above scenario in a simplified way showing only one gateway and some sensor network. Note that the RUNES LAN may involve WiFi access points as part of the tunnel infrastructure and the tunnel gateway may be mobile, connecting to the RUNES LAN via different access points at any time. This would make the sensor network a mobile network.

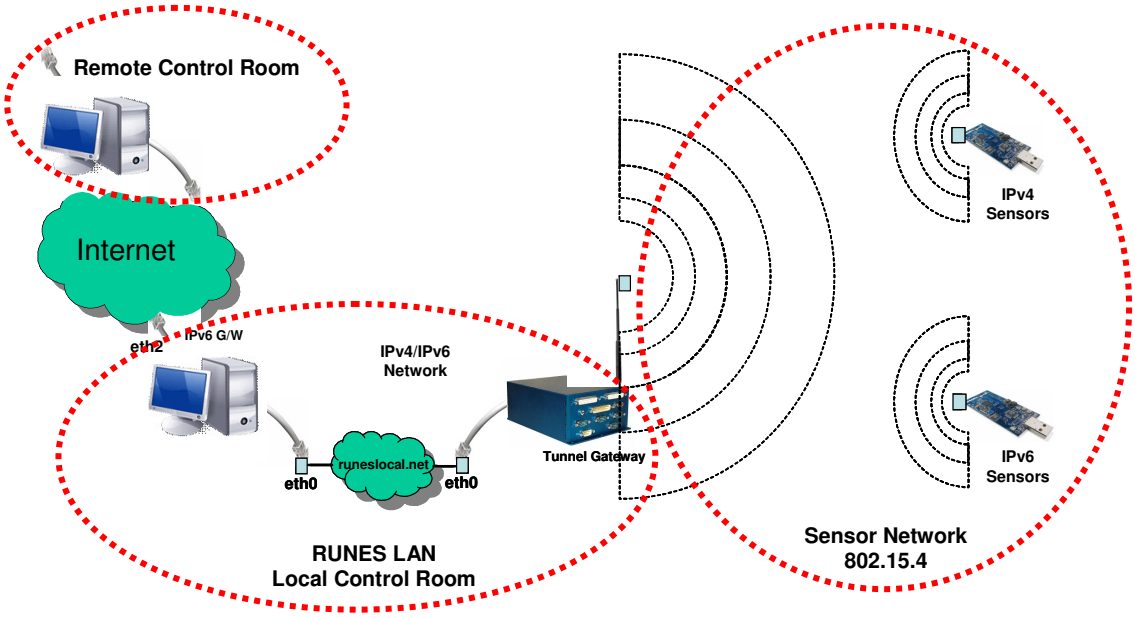

Figure 1: Schematic of demonstration configuration of RUNES IPv6 Scenario

2.2 The System to Implement the Storyline

In practice, not all of this scenario actually needs to be constructed physically in the RUNES Final IPv6 Demonstration (RF6D). The maximal scenario is that of fire fighters bringing new motes to the tunnel-on-fire scene and communicating with these motes via their mobile gateway. This is the scenario that we are implementing. The network implemented to satisfy this maximal scenario could then be adapted to capture the other case where the motes are fixed into the tunnel infrastructure, which is similar to the moving fire fighters, but does not involve a mobile sensor network.

The RF6D as described previously consists of the following equipment:

- FCR laptop: Laptop that runs the firemen control room application (which is the some as the Tunnel control room application)
- Home Agent: Laptop with Linux kernel 2.6.15, and the NEPL implementation
- Router: PC with 3 Ethernet interfaces, Linux kernel 2.6, radvd
- AP1 and AP2: Two WLAN access points
- GW: Lippert MoteMaster, 802.15.4 interface with 6lowpan extension, WLAN interface, Linux kernel 2.6.x, NEPL implementation
- SN2, SN3, and SN4: Telos TMote Sky Motes, running Contiki with 6lowpan functionality

The Tunnel Control Room (TCR) and the Firemen Control Room (FCR) are represented by a laptop. The Tunnel segment gateway or the firemen gateway is a LiPPERT Gateway [2] or a Telos Tmote flashed with the gateway version of contiki OS [3]. Sky Motes represent some of the Tunnel Infrastructure Sensor devices or sensor devices attached to firemen, measuring environmental condition (e.g.

temperatures). In reality the temperature sensors attached on the Tmote boards would not be appropriate for sensing temperatures in the range of 200°C. But the Tmote board is flexible and allows the attachment of more advanced sensors as well. Further information on these devices is given in Section 3. The tunnel network is represented by a 802.15.4 network [4].

It is not yet clear whether the specific implementation of the Lippert that we have can support both IPv4 and IPv6 on the same device; we may have two in the demonstration – one IPv4 and one IPv6.

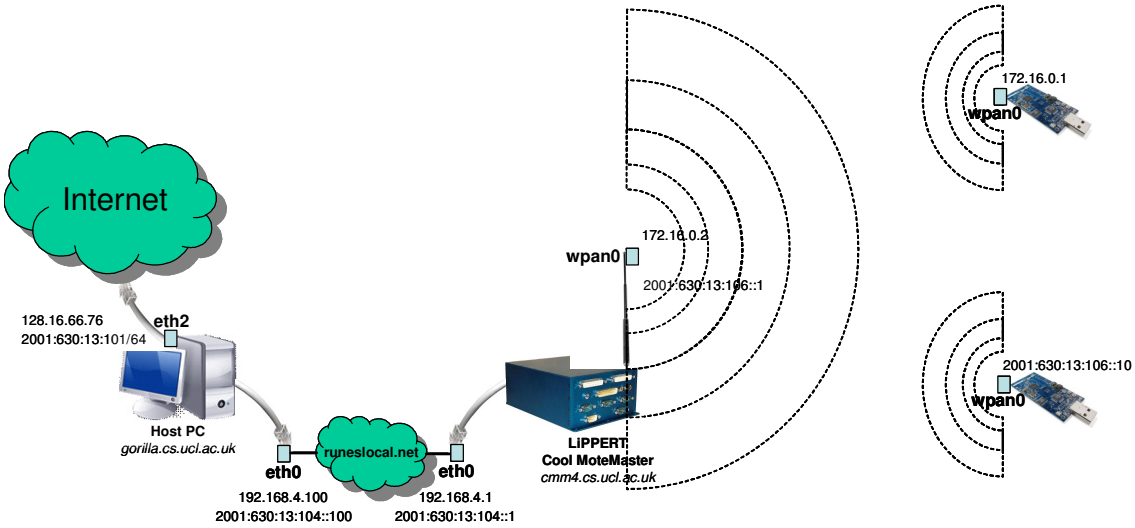

Figure 2: Implementation System of the demonstration configuration

2.3 The Functionality being implemented

Though the main activity of the IPv6 activity in the last few months of RUNES is aimed at a demonstration, the most important aspect of this activity is what is being provided.

From Figs 1 and 2, we see that there are several segments that must be developed, moving from left to right in the figure:

1. To ensure that the various functions in the control room can communicate over IPv6, and the control room analysis and display can accept data and control over IPv6 from remote stations and the tunnel gateway. This is a fairly standard activity which requires only that the control room can operate over the IPv6 stack, and that its applications have the appropriate interfaces.
2. That remote users can access the control room, both for control and data – even if they are mobile. This is again standard, provided that Mobile IPv6 (MIPv6) [5] has been implemented on the hosts.
3. To ensure that the tunnel gateway (TGW) can operate properly in the IPv6 environment. This is much more complex, and is where the real problems lie. Here the reasons and functions are the following:
   a. That the TGW can support commands and data from the Internet and the control room network via IPv6.
   b. That the TGW can run its middleware receiving its code from code servers using IPv6.

c. That the TGW can support both MIPv6 from remote users, and MANEMO [6] (remote network mobility) from mobile networks. Entities such as the emergency communications control centre and the mobile sensor networks will be of this type.
d. That the TGW can support appropriate communications with the sensor network. Here the problem is that the sensors are often resource poor, with limited addressing, memory, CPU and power capability. Thus packets must be short, and with intermittent access; with 802.15.4, the packet size is 125B. By contrast, IPv6 devices are normally expected to be always on and support both large address space and a large minimum packet size greater than 1280B. This problem has been recognised, and the 6LoWPAN protocol [7] is being developed for this purpose. One of the major activities will be to ensure that an implementation of 6LoWPAN runs on the TGW.

4. That the sensors can be addressed in either the IPv4 or the IPv6 space, can be addressed in the manner normal to such sensors, and can be auto-configured.

2.4 Storyline of the demonstration

To tie the storyline back directly to the main RUNES demonstration RFD [8], we will use a portion of the same scenario and story as used there. In order to do this, we must show the sort of figure used to illustrate the RFD scenario.

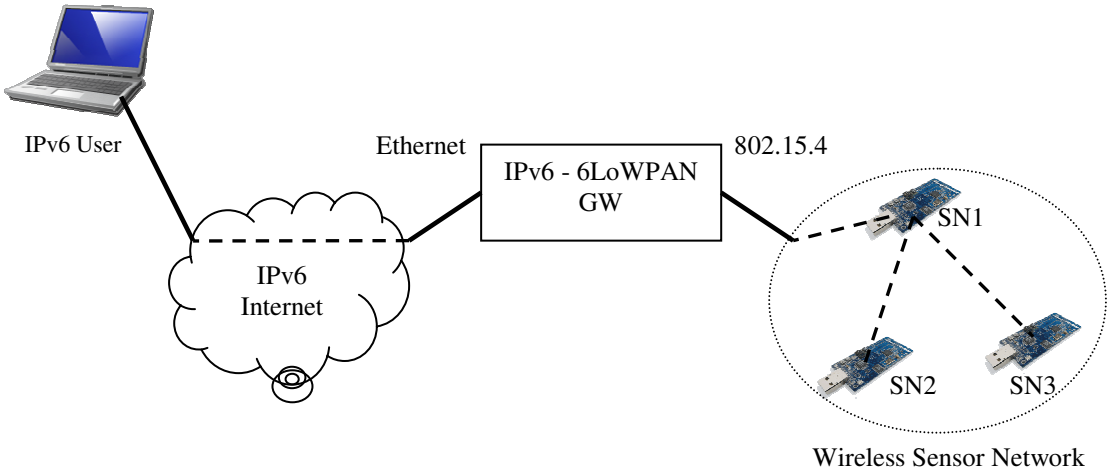

Figure 3: Internetworking scenario between 6LoWPAN WSN and IPv6 networks

The differences between Fig. 2 and Fig. 3 are slight; the latter shows a whole sensor network rather than just a sensor node. It is Fig. 3 that is implemented in practice. It shows that the Lippert gateway and the sensors have implemented both the basic IPv6 stack and the 6LoWPAN [7] architecture. The latter is a very important adjunct for wireless sensor networks that need to conserve power [8].

1. The start setup is as follows:
   - GW is attached to AP1 via WLAN
   - SN2 is attached to GW via 802.15.4
   - SN2 is autoconfigured (option b) with an address of the home prefix (to be decided)
   - NEMO functionality establishes a tunnel between GW and Home Agent.

SN2 runs the Measurement component which is configured to transmit its measurement readings using IPv6 and 6LoWPAN, via the Lippert gateway back to the Firemen Control Room.

   a. The Measurement readings are put into an IPv6 packet on SN2 and this is then fragmented according to our 6LoWPAN implementation link-layer fragmentation mechanism including compressed headers.
   b. These link-layer fragments are transmitted via 802.15.4 to the Lippert gateway where they are reassembled to form the original IPv6 packet. Debug output on the console of the Lippert gateway could be displayed to show the traversing packets and link-layer fragments
   c. The reassembled IPv6 packet on the Lippert gateway is then transmitted over the NEMO tunnel to Home Agent, where it is decapsulated and forwarded to the Firemen Control Room laptop, where the contents are extracted and the Measurement readings displayed on a very simple GUI.
   d. The fragments are sent with Robust Header Compression

2. The Firemen Control Room can also communicate with the firemen infrastructure sensors to calibrate the reading interval (how often measurement readings are taken) and critical condition (what reading level denotes 'critical' condition) thresholds.
   a. The Firemen Control Room constructs an IPv6 packet containing a calibration request (compliant with the RUNES common packet format)
   b. This IPv6 packet is transmitted through the network to the Lippert gateway, where the packet is fragmented according to our 6LoWPAN implementation link-layer fragmentation mechanism including compressed headers. Debug output on the console of the Lippert gateway could be displayed to show the traversing packets and link-layer fragments
   c. These link-layer fragments are transmitted via 802.15.4 to the sensor device SN2 where they are reassembled to form the original IPv6 packet
   d. The contents of the reassembled IPv6 packet are then extracted on the sensor SN2 and the calibration request is applied to the Measurement component
3. New sensors SN3 and SN4 are then added to the sensor network, which means new firemen join the scene.
   a. These new sensors auto-configure into the 6LoWPAN infrastructure to obtain an IPv6 address with the home prefix
   b. These new sensors also run the Measurement component and so start transmitting Measurement readings back to the Firemen Control Room where they can be visualised (as in step 1).
4. The firemen move within the tunnel and GW loses connectivity to AP1 but get connectivity to AP2. NEMO functionality in the Lippert gateway and the home agent reconfigures the tunnel so that packets from the firemen control room destined for sensor nodes are still forwarded to the Lippert gateway and from there forwarded to the respective sensor as well.

2.5 Technical Aspects of the Demonstrator

The scenario of Section 2.4 demonstrates the main aspects of the IPv6 and 6LoWPAN work that we would like to show:

- Significant portions of the demonstrator using the IPv6 protocol.
- Fragmentation/Reassembly (transmission within the sensor network)
- Header Compression (transmission within the sensor network)
- Auto-configuration (addition of new sensors).
- Network mobility (which is only available for IPv6)

Further, this is shown within the same context as the main RUNES demonstrator, where measurement readings and calibration communications are exchanged between the tunnel infrastructure sensors or sensors attached to firemen, and the Tunnel or Firemen Control Room. Thus additional components could be brought over from RUNES.

3. The Hardware and Software RUNES Components

We have many components of hardware and software available. Some of these are discussed in this section, to the extent that they are used in the RF6D demo.

3.1 LiPPERT gateways

This is the Cool MoteMaster (CMM) gateway prototype, designed by LiPPERT under RUNES, of which we have four.

The gateway offers two USB ports, three RS232 serial ports (1 console port, 2 serial ports), one Ethernet interface, one 802.15.4 interface, analogue/digital and digital/analogue converters as well as digital I/O. The full specification of the hardware is provided in the RUNES deliverable D3.4 [2].

The RUNES RFD demo used a USB hub connected to the LiPPERT gateway for communication to the Wireless Sensor Network (WSN). One could also use the 802.15.4 radio interface, which has now an updated driver from LiPPERT.

The OS running on the CMM is Linux kernel 2.6 from DENX [10]. Although DENX directly supports a number of boards, it did not directly support the PC-104 small factor PC board of the CMM and therefore the OS was slightly modified by LiPPERT. The on-board image the devices came with does not support IPv6 in the kernel. We have made the IPv6 stack to work on a CMM gateway node by cross-compiling the IPv6 DENX Linux module and flashing the updated full image to the CMM.

Unlike the Cisco routers of Section 5, the Lippert gateway has open interfaces to applications. This allows various sensor control and measurement functions to be implemented on that gateway.

3.2 Motes

We have some 60+ Telos TMote Sky sensor devices [3] available at UCL for use within RUNES; these can be added to the number of Telos TMote Sky sensor devices that other RUNES partners (including EAB and SICS) also have available. In addition to this, UCL has a mote testbed consisting of 40 Telos TMote Sky sensor devices distributed across a large open-plan office. This mote testbed forms one part of a larger Heterogeneous Experimental Network (HEN) testbed at UCL [11], capable of allowing simple control, management and execution of network experiments.

3.3 Contiki

Within RUNES, we use the Contiki Operating System [12] running on the Motes. This is a light-weight event-based operating system that has been designed at SICS specifically for low-powered and resource-constrained devices such as the motes. It currently provides an IPv4 Stack called µIP (micro IP), which although highly optimised to run on such resource-constrained devices, still remains compliant with IETF IPv4 standards. This work is scheduled to be completed for the 2nd RUNES integration meeting in Stockholm in June.

For routing, a µ-AODV implementation [13] has been provided also by SICS. This has been shown to work on the Telos TMote Sky sensor devices within RUNES and is planned for use within the RUNES Final Demonstration.

UCL has also developed one of the three RUNES Component Run-Time Kernel (CRTK) implementations of the RUNES Middleware [14]. This is the implementation written in C that runs under Contiki for the Telos TMote Sky sensor devices. The middleware provides a component-based framework that allows specific functionalities to be encapsulated within individual components, provides a mechanism for these components to interact through well-defined interface-dependency relationships, and provides functionality for the dynamic re-binding of these relationships between components and new components at runtime. The CRTK itself and the functionality that it provides is IP-version agnostic. Any components developed for the CRTK that may need to rely on underlying IP functionality can be developed also to be IP-version agnostic taking advantage of the interface provided by the underlying Contiki operating system.

3.4 µ-IP

SICS have written an 802.15.4 simulator called COOJA [15]; although this is still in an early stage (output from a Master's thesis at SICS), it is capable of emulating the functionality of motes running Contiki, and simulating experiments involving large numbers or sensor devices.

It is generally agreed that COOJA will be quite useful to help us identify possible issues with the use of 6LoWPAN in RUNES scenarios, in particular those related to a demo.

3.4.1 µ-AODV

µ-AODV [13] is to be used within the RUNES Final Demonstrator (RFD) to show how the tunnel sensor devices are able to route sensor reading data to the Tunnel Control Room, and also reconfigure by re-routing around broken sensor devices.

4. Discussion of IPv6 issues in 6LoWPAN

4.1 6LoWPAN and IPv6

Using the Gateway (TGW) of Fig. 3, the TGW on the physical/link layer transports IP packets from Ethernet to 802.15.4. The 6LoWPAN work in RUNES has defined the role of this gateway to the higher layers regarding. For example, when a User sends a request to a sensor SN1:

- The format of that request is in IPv6.

- If it is full IPv6 the request has to be transformed into 6LoWPAN in the WSN GW.
- How much state should the WSN GW maintain for this change between IPv6 to 6lowpan?

In the above scenario, the 6LoWPAN is basically only a syntactic change to an IPv6 packet. The mechanisms to perform this syntactic change has to exist on the WSN GW as it should be transparent to the User if a node uses 6LoWPAN or regular IPv6.

Regarding the state, there has to be some way to resolve an IPv6 address into a 802.15.4 address. If this is done by a directory look-up or a mechanism (for instance turning the IPv6 address into a 802.15.4 MAC-address by removing the prefix), determines whether state information is required. The 6LoWPAN specification is not clear on how this should be done but at least proposes a method on how to create an IPv6-address from a 802.15.4 MAC-address. A state is anyhow required for the mesh-protocol. It is also assumed that link-layer addressing and routing, such as μ-AODV, is in operation in the WSN network.

Security is one more factor that may affect the overall architecture of the WSN islands and the Internet. As 6LoWPAN is a first step towards efficient IPv6 packet transport over 802.15.4, we have not implemented any security yet at this point; however, we must consider that the gateway may become a security association endpoint.

4.2 PAN ID coordinator selection

There is little information about the PAN coordinator selection mechanism. It does not seem to be a mechanism that has been decided upon, yet there are mentions of PAN coordinator selection in mailing lists and drafts but no actual mechanism description in a draft form.

The existing discussions only mention that the PAN coordinator should be selected and that a PAN coordinator should be a Full Function Device (FFD) - i.e., one that can act as a gateway between the PAN and the outside world.

A possible mechanism for PAN coordinator selection could be as described in [1]:

"…

> All networks must have one and only one PAN Coordinator. This must be an FFD (Full Function Device). The selection of the PAN Coordinator is the first step in setting up an IEEE 802.15.4 based network. The PAN Coordinator can be selected in a number of ways:
>
> - In some networks, there may be only one device that is eligible to become the PAN Coordinator; for example, networks with only one FFD or in which a particular device has been designed to be the PAN Coordinator (for example; the device that acts as the gateway to the outside world).
> - In networks with more than one FFD, it may be the case that any of the FFDs can act as the PAN Coordinator. In this case, the user may or may not wish to pre-determine which device becomes the PAN Coordinator:
>
> - The user may determine the FFD that is to become the PAN Coordinator through some action, such as pressing a button.

- It may not matter which FFD becomes the PAN Coordinator and the choice can be left to chance; for example, by having all the FFDs perform an Active Channel Scan and by assigning the PAN Coordinator responsibility to the first device that returns a negative result (no other PAN Coordinator detected).

Once the PAN Coordinator has been established, a PAN ID must be assigned to the network. It is possible to decide and fix the PAN ID in advance. However, care must be taken, as the PAN ID must be different from that of any other network that can be detected in the vicinity. Normally, the PAN ID is assigned by the PAN Coordinator, taking into account the PAN IDs of any other PAN Coordinators that it can hear.

… "

The 6LoWPAN group of the IETF has recently been discussing re-chartering, where the issue of PAN coordinator selection may fit as a working item dealing with the interface of a WSN to the outside world.

From a RUNES perspective, in a tunnel scenario we have many sensors and gateway devices through the tunnel, and so several paths/connections to the outside world through the multiple gateways (assuming that some gateways within the tunnel are also wired directly to the outside instead of all going through the tunnel end-point gateways).

One unresolved question is whether there is one PAN coordinator for the whole tunnel network or it is split between tunnel segments. Depending on reachability between devices and tunnel segments, there may be several 'PANs' in this context. In the general case, this needs further study – and possibly extension to the implementation. How do we account for this on an overall tunnel scale for a communication interface to the outside world? What if there are multiple routes to the outside world? (The latter may imply multi-homing and access selection that is something that 6LoWPAN is not concerned with.)

The 6LoWPAN draft assumes a working 802.15.4 link layer while we do not have such a thing on Contiki/Telos. Thus, having multiple PANs is not a problem related to 6LoWPAN directly, but rather is an 802.15.4 problem. The fact that we do not have a full-blown 802.15.4 protocol on Contiki/Telos does not mean that we have to implement it.

Having a node belong to multiple PANs introduces further issues for 6LoWPAN relating to fragmentation/reassembly and header compression. A sensor node should remain in one PAN for the duration of at least one IPv6 packet (assuming 1 IPv6 packet = N 6LoWPAN fragments) and for as long as it has some fragmented packet on the fly. Otherwise if different fragments from a sensor node fly to different PAN coordinators (on their way out to the Internet) then there is no way for reassembling the packet.

At this stage we can assume that the 802.15.4 link layer has all its parts working (different PAN coordinators, PAN members connect to the PAN coordinator, they are aware of its address etc). So for 6LoWPAN testing purposes we can make the following assumptions:

- assume that the nodes know the PAN ID

- the nodes have a short (16-bit) address after an association event with the PAN coordinator, and
- the coordinator is the second radio of a gateway (i.e.: LiPPERT) or a mote connected to PC.

5. The Relation of the RUNES Demonstration to U-2010, WP4

5.1 D1, D3, D5 and D9 - RUNES Sensor Net + MARS

Currently in the RUNES demonstration, there is a sensor network, to which we will add a camera. This sends a sensor and video stream, controlled from the Lippert G/w, which includes an Ethernet.

We propose under U-2010 to be able to send the data from the sensor network through to a Mobile Network controlled from the MARS router in a van, and to control the Sensor network from consoles in the van through the MARS. We would also like to relay the data to other WANs including UMTS and the SES-ASTRA emergency satellite network.

The link between the satellite network and the MARS has just been completed by Cisco and SES-Astra, but we do not have yet any information on the performance. IPv6 support on this link is underway, but this will only be in the form of IPv6-over-IPv4 at the moment. The IPModem used by AstraConnect is based on Linux and one would expect a full IPv6 stack to be possible to integrate, but this does not involve UCL at all.

The subproject described above may require that we can use streams of IPv6 traffic, with Mobile IP and 6LoWPAN. The Mobile IP may well be NEMO.

5.2 Cisco Mobile Access Routers (MARs)

We were provided with 4 MARS stacks from Cisco. These were the 3250s, without cables or enclosures. Lancaster supplied us also the cables – but no rugged enclosures. We discuss below the configurations that we have. They are not used in the RF6D, but would be used in the U-2010 version.

5.2.1 Hardware

We currently have 4 Cisco 3250 MARS routers with the following boards:

- Mobile Router Power Card (MRPC: power unit card)
- Mobile Access Router Card (MARC: main routing board)
- Serial Interface Card (SMIC: provides 4 serial ports incl. console)
- Fast-Ethernet Switch Mobile Interface Card (FESMIC: provides 4 x 10/100 FastEthernet ports, and one Serial port)
- Wireless Mobile Interface Card (WMIC: 802.11 b/g with 2 antennae)

Whilst the MARC integrates with the FESMIC and SMIC cards directly, the WMIC runs independently and only draws power from the PC104 bus. The WMIC is interfaced to the MARC via Ethernet only.

All necessary cables and power supplies have been provided to us by Lancaster U. Note we only have the bare MAR stacks and cables, no rugged enclosures.

5.2.2 Software:

The MARs are currently running two independent IOS images. The WMIC runs a production version of IOS without IPv6 features, whilst the MARC runs an experimental Cisco Advanced Enterprise Image that implements Cisco enhanced NEMO and MANEMO. These have been flashed to the latest IOS.

	Firmware filename	Date obtained by UCL
WMIC	???	March 2005 (provided on-board)
MARC	c3250-adventerprisek9-mz	October 2006

The MARS software is IPv6 aware. The WMIC runs a different IOS image, which is not v6-aware. It can be configured as a bridge that just forwards IPv6 packets on to its Ethernet interface (which was our setup for 6NET); one of the MARs serves the 7th floor in the CS Building as an Access Point to our IPv6-enabled LAN.

We have a couple of scripts from Cisco which demonstrate basic MANEMO functionality. We can reproduce this in our Lab and do basic debugging using the Ethereal parser lib with Tree Discovery (ethereal-0.10.14-1.i386.rpm by Teco Boot provided to us by Cisco).

References

[1] U2010 fire-in-the-tunnel video

[2] http://www.ist-runes.org/docs/deliverables/D3_04.pdf

[3] http://www.willow.co.uk/html/wireless.html

[4] 802.15.4

[5] MIPv6

[6] MANEMO

[7] http://en.wikipedia.org/wiki/6loWPAN

[8] http://www.afrasiabi.com/draft.htm

[9] Final RUNES Demonstrator

[10] http://www.denx.de/wiki/DULG/ELDK

[11] HEN

[12] CONTIKI

[13] Micro-aodv

[14] RUNES Middleware

[15] COOJA