Software Radio Platform for Network-Assisted Device-to-Device (NA-D2D) Concepts

Vicknesan Ayadurai
Ericsson Research
Färögatan 6
164 80 Stockholm
Sweden
vicknesan.ayadurai@ericsson.com

Mikael Prytz
Ericsson Research
Färögatan 6
164 80 Stockholm
Sweden
mikael.prytz@ericsson.com

ABSTRACT

This paper describes a small-scale software radio platform that has been developed and implemented to investigate the new mobile network concept of Network-Assisted Device-to-Device (NA-D2D) communications. The implementation includes mechanisms for mode selection, direct-link quality estimation, and resource allocation. We show that significant benefits were reaped from this real-world implementation despite the relatively simplistic scenario emulated by our indoor laboratory setup. Apart from the obvious advantages of testing the concept over a real wireless medium with its associated physical characteristics, other unforeseen factors were also uncovered. Asymmetries in radio channels, variations in supposedly identical node hardware, and timing issues, were just some of the experiences encountered and fed back into the research design process which resulted in significant refinements and improvements made to the mechanisms and algorithms of the NA-D2D communications concept studied.

Categories and Subject Descriptors

C.2.1 [**Computer-communications networks**]: Network architecture and design—*wireless communication.*

General Terms

Algorithms, Measurement, Experimentation

Keywords

Software radio, mobile networks, device-to-device, WARP, FPGA

1. BACKGROUND AND INTRODUCTION

The recent uptake of mobile broadband has seen tremendous increase in data traffic in todays mobile networks. As we evolve towards a more networked society, the sheer volume of data, as well as number and new *types* of devices connecting to the network will increase immensely. All of this puts an increased strain on the limited radio spectrum available as well as on the networks supporting these devices. As such, alternative wireless concepts need to be investigated and developed.

While commercial cellular mobile networks of today conform to strict standardized specifications, the real challenge in their design, implementation and deployment arise when the number of nodes or devices are large, resulting in traffic volumes which become demanding to handle. From this aspect a small-scale software radio platform is an inadequate tool to realistically emulate a commercial mobile network system.

Nonetheless, we would argue that other benefits could still be derived from a small-scale prototyping activity. While the sheer large-volume challenges may not be realistically studied, the feasibility of specific new mechanisms and protocols could be evaluated, under the conditions of a physical radio channel. Unforeseen real-world issues would be detected and potentially resolved. Finally, the value of a live demonstration to better illustrate and convey a new concept to a larger target audience should never be underestimated.

To this end a small-scale software radio platform enables practical experimentation at an early research stage, while at the same time incurring relatively low cost and effort.

1.1 Network-Assisted Device-to-Device concept

In traditional mobile networks all communication takes place via a network node, typically the base-station (BS). Two devices wanting to exchange information would first need to transmit their data up to the network, which in turn would then send it down back to the recipient. This would occur regardless of whether the two devices in question were located at opposite ends of the planet or if they were physically adjacent to each other in the same radio cell.

With NA-D2D communications the network would detect the presence of an *intra-cell* data exchange and could therefore potentially instruct the two devices to transfer their information directly to each other, bypassing the network altogether. The transfer would still take place utilizing the network-controlled radio spectrum, but would require less power and fewer transmissions for the transmitting devices, which at the same time would benefit from higher throughput and lower delays, all due to the close proximity of the two devices[3].

SIGCOMM'13, August 12–16, 2013, Hong Kong, China.

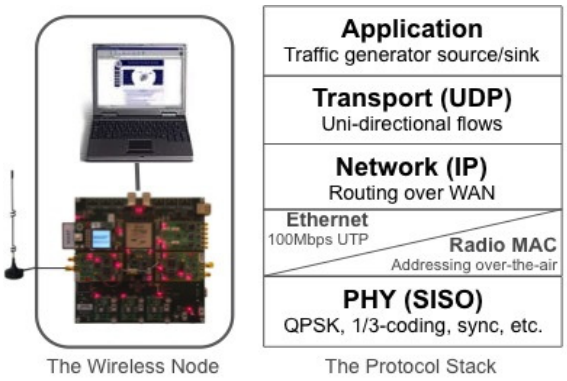

Figure 1: Our wireless node comprising of a WARP-board tethered via a dedicated 100Mbps ethernet link to a Unix PC, and the corresponding protocol stack

Intra-cell data transfer scenarios will become more prevalent with the arrival of new device types into the networks of the future, and as such, the NA-D2D communications concept is being actively discussed within standardization forums like 3GPP[1].

2. SOFTWARE RADIO WIRELESS NODE

The wireless nodes we utilize for our experiments comprise of two main components: a Wireless Open-Access Research Platform (WARP) board[5][2], tethered via a dedicated full-duplex 100Mbps ethernet link to a Unix[1] PC. Both of these are standard items available off-the-shelf. Figure 1 shows the physical representation of the wireless node comprising of these two components, and the corresponding protocol stack view.

2.1 The Wireless Open-Access Research Platform (WARP) board

The WARP-board we employed consisted of a Xilinx Virtex II Pro FPGA chip, and surrounding peripheral components. Using Matlab, Simulink and Xilinx's System Generator, we implemented a custom-built SISO OFDM radio physical layer (PHY) with QPSK modulation and a 1/3-rate convolutional coder and corresponding Viterbi decoder. This radio chain runs on the FPGA chip, and is connected to an analog radio "daughter-card" physically attached to the WARP board. The chipset on this daughter-card enables radio transmissions in the Industrial, Scientific and Medical (ISM) bands. In our lab we configured this card for operation at a center frequency of 5580MHz.

Additionally, we added an over-the-air time synchronization mechanism which enabled all wireless nodes to synchronize to a beacon or pilot signal generated by a single "Master" wireless node. With this synchronization capability we then built a TDMA overlay consisting of 10ms time-frames, comprising of ten 1ms timeslots each.

[1]The main criteria in the selection of operating system is the ability to configure networking and routing functionality for the PC flexibly. As such, while FreeBSD has been our O/S of choice, we have also successfully tethered our boards to other Unix variants namely Ubuntu Linux, and Mac OSX.

Figure 2: Screen-shot of the `warp_drive` application run on the wireless node's PC. The top-portion displays statistics, while the remainder shows system log messages received from the board.

The Virtex II Pro FPGA also has dual on-board PPC processors, memory and data plus control busses, upon which we ran custom-developed software written in the C programming language, implementing a radio medium-access control (MAC) protocol[2] and other radio-network related control mechanisms.

Finally, a software control module was also implemented to enable user-interaction with this wireless node via an ethernet connection.

2.2 The Unix PC

The PC tethered to the WARP-board implements the higher-layer protocols for our wireless node. By customizing the routing as well as address resolution protocol (ARP) tables, all traffic will leave the PC via the dedicated ethernet link to the WARP-board.

Traffic generated by applications on this PC will leave via this ethernet link to the WARP-board, where it will then be transmitted out over the air via the SISO radio implementation. Conversely, packets received via this radio interface on the board will be pushed out the ethernet link up to the PC. Furthermore, if the PC has a second ethernet interface connected to a wired network, then with some careful addressing and routing configuration, it is possible to stream *network* traffic over the air via this wireless node.

Apart from the user-plane traffic described above which flows "through" the WARP-board, a control-plane feature is also implemented for communications between the board and the PC. An in-house developed terminal-emulation type application called `warp_drive` (See Figure 2) is run on the PC which receives statistics and log messages from the board. In the other direction, keystroke input to this program is sent out to the board where it is then interpreted by this software control module running on the FPGA. This way bidirectional user-interaction with the WARP-board is possible.

[2]Our extremely simple radio MAC protocol provides addressing functionality only, and hence, requires no feedback channel or additional buffers for ACKs or retransmissions, etc.

3. IMPLEMENTATION OF NA-D2D COMMUNICATIONS CONCEPTS

In this section we describe the NA-D2D communications concept and mechanisms we wish to study, and how this was emulated using the software radio wireless nodes described in section 2 above.

One of the numerous benefits of device-to-device communications are savings in radio resources[4]. Intuitively, one can imagine less radio resources being consumed in transferring data directly between two devices, i.e., *sending device* → *receiving device*, compared to via an intermediate third device, i.e., *sending device* → *base-station* → *receiving device*.

As mentioned in section 2.1, we developed a time synchronization mechanism to our software radio implementation enabling us to emulate a TDMA setup with 10-timeslot frames. Hence, the radio resources within our emulated system here would be **timeslots**.

The scenario we would thus like to study is how to reduce the system's consumed resources, i.e., timeslot usage, by switching from traditional cellular operation via a BS (henceforth, "classic-mode") to direct device-to-device operation (henceforth, "D2D-mode") in a reliable and efficient manner. In our scenario we assume that two devices initially exchange information in "classic-mode" and are subsequently switched over to "D2D-mode" by the network when it decides that this is possible, hence, *Network-Assisted* D2D.

3.1 Emulated network setup

Let us consider a simple one-cell TDMA-based cellular mobile system comprising of a single base-station node, *BS*, representing the entire network infrastructure[3], and two end-user devices, imaginatively dubbed *LEFT* and *RIGHT*.

All three entities in our scenario are emulated by the wireless node described in section 2. Software configuration differentiates their behaviour and enables *BS* to execute network functionality. This includes transmission of the periodic synchronization beacon every 10ms, as well as handling radio resource allocation.

In our setup Timeslot 0 and Timeslot 5 of the 10-slot TDMA frame are reserved as Downlink and Uplink Control Channels (DCC and UCC) respectively. Resource allocation thus involves assigning uplink(UL) and downlink(DL) timeslots to newly arriving devices from the system's remaining pool of 8 timeslots.

Nodes *LEFT* and *RIGHT* execute identical device software, and listen out for a beacon in order to synchronize to the network, and subsequently, expect to receive instructions from the network on the DCC.

In "classic-mode" operation, each device would require two user-data timeslots each to connect to the network: an uplink timeslot and a downlink timeslot. Hence, for information to be exchanged between the two devices via *BS*, a total of four system timeslots will be consumed in our cell for user-data.

In "D2D-mode" operation the uplink timeslot of device *LEFT* could be also assigned by the network as the *downlink* timeslot of the device *RIGHT*[4], and similarly, vice-versa. This would result in transmissions going directly between the two devices, bypassing *BS* altogether, and consuming only two timeslots in total; one timeslot for the *LEFT*→*RIGHT* traffic and one for the *RIGHT*→*LEFT* traffic.

[3]Since *BS* represents the base-station *and* the entire infrastructure network, we henceforth use the terms "base-station" and "the network" interchangeably

[4]This applies to scenarios where transmitted packets from *LEFT* is destined for *RIGHT* of course.

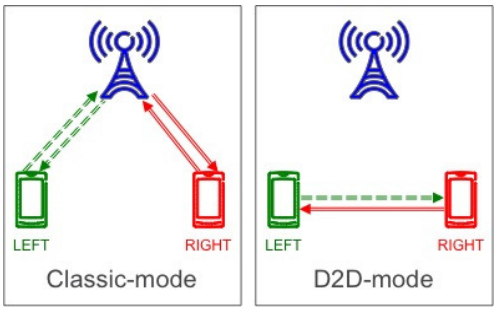

Figure 3: Two views of our 3-node single-cell emulated mobile network, where each user-data timeslot is indicated by arrow-lines. The left view shows "classic-mode" operation where four system timeslots are utilized for the two devices to exchange data. The right view shows the devices in "D2D-mode" operation whereby only two timeslots are utilized.

Figure 3 illustrates the "classic-mode" scenario on the left, and the "D2D-mode" scenario on the right.

However, D2D-mode is only possible if the two devices are in sufficiently close proximity or more accurately, have a sufficiently good direct radio-link between them. The challenge therefore is how the network can learn about the quality of this direct link between the two device nodes in order to make the decision on which mode of operation should be used for this intra-cell communication between *LEFT* and *RIGHT*.

3.2 Emulated "classic-mode" operation and resource allocation

We investigate a scenario where the two devices, *LEFT* and *RIGHT*, wish to exchange information between them. Upon connecting to the network they transmit a continuous stream of data to each other via *BS*. The network detects that there is an ongoing intra-cell traffic flow and initiates a mechanism to investigate the quality of the direct link between *LEFT* and *RIGHT*. If the outcome of the investigation is positive then the network can reassign timeslots, such that the information transfer between the nodes now takes place directly in D2D-mode, bypassing the *BS* altogether and saving resources consumed, i.e., switching from the left image to right in Figure 3.

The sequence of events where the two devices "attach" to the network is as follows:

1. The *BS* node is up and running. It transmits a beacon every 10ms and thereby has an awareness of the TDMA-frame being used in the system. It monitors Timeslot 5, the Uplink Control Channel (UCC) slot for incoming device requests, and is ready to transmit messages on the Downlink Control Channel (DCC) if necessary. At this point two of the system's 10 timeslots are already used by the control-channels.

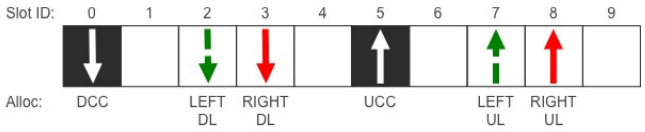

Figure 4: Graphical view of the system's TDMA frame slot allocation after the successful "attach" operation of devices *LEFT* and *RIGHT* to the network.

2. Device *LEFT* is switched on and enables its receiver continuously[5]. Upon detecting the beacon from *BS*, device *LEFT* establishes the time-frame being used and is synchronized to the system. Node synchronization is handled automatically by the hardware whenever a beacon is detected. However, at this point the node hasn't been assigned any uplink or downlink data timeslots yet.

3. Via the `warp_drive` terminal-emulation program running on *LEFT*'s PC, we manually issue an `ATTACH`-command which is transmitted out the UCC. *BS* detects this incoming control-message and allocates one timeslot for device *LEFT*'s uplink and one timeslot for its downlink from the pool of available timeslots. It then configures these timeslots locally (i.e., at *BS*).

4. Next, a control-message with these two slot allocations is then transmitted by *BS* on the DCC, which is always monitored by the synchronized device-nodes. This is detected by device *LEFT*[6], which then configures its two slots locally, as per the instructions of the control-message received. At this point an additional two timeslots are consumed resulting in a total of four of the system's 10 timeslots being used up.

5. Next, device *RIGHT* is now turned on, and it undergoes the same attachment-procedure as device *LEFT* above, after which an additional two timeslots are consumed.

Upon completing the above steps we now have a TDMA "cellular system" with two devices up and running where six of the 10 system timeslots are allocated. Figure 4 illustrates this final system resource allocation in place.

We can now start a traffic-generator application on device *LEFT*'s PC and send a continuous stream of packets bound for device *RIGHT*. Since our traffic-generator uses UDP and the radio MAC has no retransmissions or acknowledgements, this stream will be uni-directional from *LEFT* to *RIGHT*.

The data packets flow from device *LEFT*'s PC to device *LEFT*'s WARP-board via the ethernet link, then over the air on device *LEFT*'s uplink timeslot, Slot 7. These packets

[5]When the wireless node is synchronized the hardware design switches off the radio when it is not needed. Hence, the radio receiver is turned on only on the timeslot(s) where the wireless node expects to be receiving information. Similarly, the radio transmitter is only enabled for timeslots when there is data for transmission.

[6]The DCC and UCC are broadcast channels. As such the radio MAC addressing in the packet-header is used for receivers to distinguish the validity and relevance of a received control-message packet on these channels

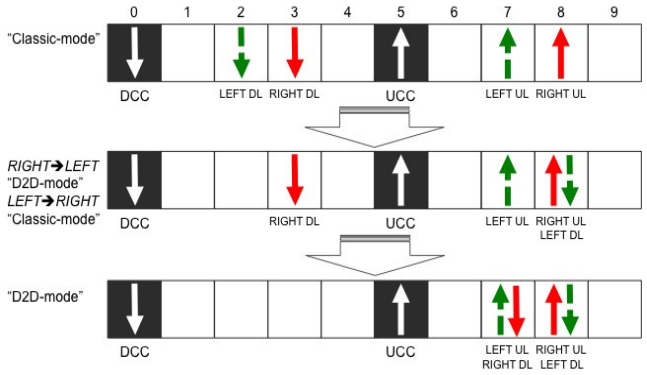

Figure 5: Evolution of the system resource allocation, from "classic-mode" operation (top), to the intermediate stage of "D2D-mode" for *RIGHT*→*LEFT* direction only (middle), to full bidirectional "D2D-mode" operation for data transfer between our two devices (bottom).

are received by *BS*'s WARP-board. Based on the addressing in the packet's radio MAC header, *BS* transmits this received packet back out its radio interface on device *RIGHT*'s downlink timeslot, Slot 3. Device *RIGHT*'s WARP-board receives these packets from *BS*, which are then sent out the WARP-board's ethernet link to *RIGHT*'s PC, where they are received by a corresponding traffic-generator[7] sink program.

A corresponding traffic-generator application may also be started on device *RIGHT*. This will provide a second unidirectional stream in the opposite direction, giving us full-duplex bidirectional traffic flows between *LEFT* and *RIGHT*, via *BS*.

3.3 Emulated transition to "D2D-mode"

Having established "classic-mode" operation for our two devices, and detecting[8] an intra-cell traffic pattern between them, the next step would be for the network to investigate the feasibility of "D2D-mode" operation for devices *LEFT* and *RIGHT*.

We propose a simple mechanism which utilizes the existing hardware capabilities of the wireless nodes, and introduces minimal changes to the software operation of our setup.

The mechanism we propose is illustrated visually with slot allocations in Figure 5, and works as follows:

1. The network discovers ongoing intra-cell traffic between two of its devices, and it now needs to evaluate the quality of the direct channel between these two devices.

2. The network arbitrarily selects one of the two nodes, say *LEFT*, and sends a new downlink control-message

[7]The traffic-stream generated out the PC's wired interface consisted of 94-byte packets (made up of an ethernet header, an IPv4 header, a UDP header and a payload) which was transmitted every 100ms. When transmitted out of the radio interface an additional 12-byte radio MAC header was appended, resulting in 106-byte packets sent over the air.

[8]Detection of the presence of ongoing "intra-cell" traffic is assumed possible by the network via mechanisms such as deep packet inspection (DPI), etc., and is not described here.

to it, instructing it to "eavesdrop" on device *RIGHT*'s *uplink transmission*, i.e., SLOT 8, for a specified short period. *LEFT* should record both the radio MAC sequence number (SN) and receiver signal-strength indication(RSSI) level for each packet it overhears on this slot. These samples, $SAMPLE_{LEFT}$, are then to be reported back to the network via a newly defined uplink control-message.

3. At the same time, *BS* itself also records radio MAC SN and RSSI values it receives from *RIGHT*'s uplink transmission on SLOT 8, let's call this $SAMPLE_{BS}$.

4. A short while later *BS* receives a sample report from device *LEFT*'s "eavesdropping" campaign via the UCC, $SAMPLE_{LEFT}$.

5. By comparing the received $SAMPLE_{LEFT}$ and its own $SAMPLE_{BS}$ the network can determine:

   - the percentage of packets successfully received by device *LEFT*, via listening directly to device *RIGHT*'s uplink timeslot, and
   - the relative separation between *RIGHT*→*LEFT* from the reported RSSI values.

   which together, when combined, provides an indication of the quality of the *RIGHT*→*LEFT* direct-channel conditions.

6. If the evaluation of the two sample reports above are "positive"[9] the network can issue a new control-message to device *LEFT*, reconfiguring its downlink timeslot to that of device *RIGHT*'s uplink timeslot (Slot 8 in this case). This frees the slot originally assigned for *LEFT*'s downlink traffic during the initial "attach" process described in sub-section 3.2, i.e., Slot 2. This slot re-allocation is depicted in the top-to-middle transition in Figure 5.

7. Note that there is no change to device *RIGHT*'s configuration whatsoever and in fact, *RIGHT* is completely unaware that it is now transmitting directly to device *LEFT* instead of to *BS*.

After executing the above steps we are in an intermediate stage where traffic flowing from *RIGHT*→*LEFT* takes a direct path while traffic flowing from *LEFT*→*RIGHT* still operates in "classic-mode" via the network.

The above steps need to be repeated in the opposite direction in order for *LEFT*→*RIGHT* traffic to also flow directly, thereby resulting in full-duplex "D2D-mode".

The middle-to-bottom figures in Figure 5 illustrate the new slot allocations upon repeating the process for the opposite direction. Overall, Figure 5 above illustrates the system resource allocation reduction from the original six timeslots when operating in "classic-mode" down to four timeslots in full-duplex "D2D-mode" operation.

[9]The criteria for the definition of "positive" could be tuned according to requirements and end-user expectations, etc., and will not be considered here.

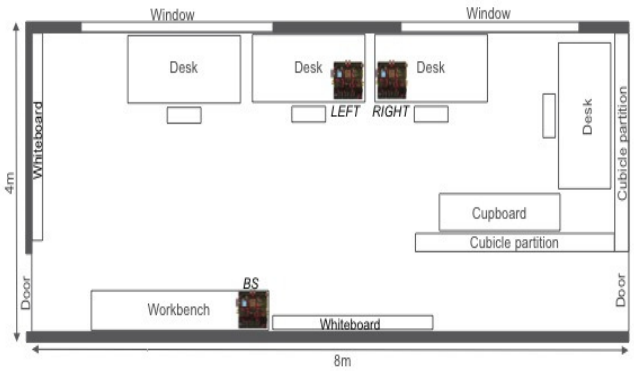

Figure 6: Floor plan of our indoor lab-room where the measurements were carried out, showing the location of the nodes involved.

4. MODE-SELECTION MEASUREMENTS AND RESULTS

While much knowledge was obtained from the overall implementation work, we will focus on the experiences gathered specifically from the mode selection mechanism investigated, as outlined earlier in section 3.3.

Our measurements and experiments were conducted in a rectangular-shaped indoor lab room with standard office furnishings. The wireless nodes *LEFT*, *RIGHT*, and *BS* were positioned at desk-height, and formed a triangle. A floor plan of this setup is depicted in Figure 6.

The boards were powered-on and "classic-mode" operation was established as described in section 3.2. A continuous bi-directional traffic stream between the two device-nodes was then initiated, which flowed via the network. Mode-transition from "classic-mode" to "D2D-mode", as described in section 3.3, was then initiated by *BS* via a command on its PC's `warp_drive` program.

In the experiment setup there was no power-control mechanism in place. All wireless nodes were configured to transmit packets out the radio interface at a fixed constant transmit power of 15dBm (≈32mW).

Received power, RSSI, is available from our SISO implementation on the WARP-board as a scalar 10-bit value. For the interested reader, the relationship between this 10-bit RSSI value and *dBm* is given by the equation:

$$dBm = -100 + \left(\frac{70}{1023} \times RSSI_{10\text{-}bit}\right) \quad (1)$$

However, we will consider only the scalar 10-bit RSSI value in the following discussion.

The measurements were repeated for different conditions of the wireless nodes in our lab, and the results of the measurement reports, $SAMPLE_{BS}$ and $SAMPLE_{RIGHT}$ (or in some cases $SAMPLE_{LEFT}$) examined. Some of the interesting mode-selection exercise scenarios are:

- Good direct-channel - devices in close proximity
- Bad direct-channel - device separation is larger than device-to-*BS* separation
- Timing issues - "real-world" causes algorithm failure

- Asymmetries - Hardware and/or unequal direct-link radio channels

4.1 Good direct-channel

Below is the result of *RIGHT* "eavesdropping" on *LEFT*'s uplink transmission. The following is the `warp_drive` console output at the network-node, i.e., *BS*:

```
D2D slot 7 direct-mode measurements:
 =>Rx SN:  80  81  82  83  84  85  86  87  88  89
 =>RSSI : 179 189 181 176 185 184 181 182 190 180
D2D_CTRLMSG_RPT_MEAS received from Node 9 @ FID=24
D2D slot 7 direct-mode measurements:
 =>Rx SN:  80  81  82  83  84  85  86  87  88  89
 =>RSSI : 291 283 292 288 288 299 283 296 284 294
```

The first two rows of numbers shows the network's Slot 7 measurements at *BS*, i.e., $SAMPLE_{BS}$. These consist of the radio MAC sequence number (`Rx SN:`) and the received signal-strength indicator (`RSSI:`) of 10 packets received on Slot 7.

Below this we see a second similar two rows of numbers. These are also measurements made on Slot 7, but as seen by device *RIGHT*, i.e., $SAMPLE_{RIGHT}$, and reported back to *BS* via the UCC.

Comparing the sequence numbers of the two samples, we see that both *BS* and *RIGHT* saw the same packets transmitted by *LEFT* on its uplink slot, Slot 7.

However, we observe that the RSSI values of the two samples differ quite significantly. *BS*, which is placed approximately 3 meters across the room from *LEFT*, reports RSSI values of around 180. *RIGHT* however, which is just adjacent, i.e., ∼0.25m, to device *LEFT* (see Figure 6), reports significantly stronger RSSI values of just below 300 .

Based on these two sample reports above the network would conclude that a direct *LEFT*→*RIGHT* transmission would be feasible, and subsequently reconfigure the DL timeslot of device *RIGHT* accordingly.

4.2 Bad direct-channel

In this experiment setup we moved device *RIGHT* away from device *LEFT*, introducing a larger physical separation between them, and resulting in the three wireless nodes forming an (almost) equilateral triangle. After executing the "eavesdropping exercise" as before, *BS*'s `warp_drive` console output is shown below:

```
D2D slot 7 direct-mode measurements:
 =>Rx SN:  12  13  14  15  16  17  18  19  20  21
 =>RSSI : 180 170 174 166 177 182 179 176 165 179
D2D_CTRLMSG_RPT_MEAS received from Node 9 @ FID=0
D2D slot 7 direct-mode measurements:
 =>Rx SN:  16  17  18  19  20  23  24  25  26  27
 =>RSSI : 141 134 137 132 137 141 141 140 144 134
```

We observe that the $SAMPLE_{BS}$ RSSI values are still approximately the same as before, i.e., around 180, since *BS* and *LEFT* haven't moved. However, we observe now that the $SAMPLE_{RIGHT}$ RSSI values are significantly lower than before at around 140 (i.e., as compared to 300 in the 'Good direct-channel' measurements in section 4.1).

More critically, we observe that $SAMPLE_{RIGHT}$ SNs observed do not match the $SAMPLE_{BS}$ SN series. Furthermore, we also note that these SNs are not contiguous, indicating that not all transmitted packets by *LEFT* were detected by *RIGHT* on the direct *LEFT*→*RIGHT* link.

Based on this outcome, the network would conclude that "D2D-mode" operation is *not possible* under these conditions, and subsequently, no timeslot reconfiguration needs to take place for the device nodes.

4.3 Timing issues

One of the beneficial outcomes derived from the prototyping activity is the effect of real-world timing issues on the algorithm. Consider the following *BS* `warp_drive` console output:

```
D2D slot 7 direct-mode measurements:
 =>Rx SN: 112 113 114 115 116 117 118 119 120 121
 =>RSSI : 188 180 186 183 174 190 192 180 187 181
D2D_CTRLMSG_RPT_MEAS received from Node 9 @ FID=59
D2D slot 7 direct-mode measurements:
 =>Rx SN: 113 114 115 116 117 118 119 120 121 122
 =>RSSI : 296 300 286 305 297 281 299 293 301 293
```

In this experiment we see that the RSSI values of the direct link from $SAMPLE_{RIGHT}$, averaging around 300, is significant stronger than the $SAMPLE_{BS}$ values of approximately 180. This should be a clear indication that "D2D-mode" operation should be possible.

However, we observe that the two sets of SNs reported, even though contiguous, are not *identical*.

This can be attributed to timing delays. The network instructs *RIGHT* to start monitoring Slot 7 via a control message sent over the DCC, and subsequently starts monitoring Slot 7 itself. The network detects packet SN 112 as the first packet. However, due to inherent delays in the system, *RIGHT* commences monitoring SLOT 7 only *after* SN 112 has been transmitted by *LEFT*, missing it completely. As a result SN 113 is the first packet *RIGHT* sees on SLOT 7.

In the first implementation of this mechanism the above sample analysis erroneously resulted in a failure to switch to direct-mode for the *LEFT*→*RIGHT* direction, simply because the two sets of SNs were not identical. Knowledge on this potential timing issue has since been fed back into the algorithm design process, resulting in an updated implementation which now anticipates and handles this condition correctly.

4.4 Asymmetries in real-life

Finally, in this experiment we have the scenario where *LEFT* and *RIGHT* are operating in "classic-mode", and are both sending packets over the air via their respective UL slots, with the same fixed constant transmit-power.

The result of *LEFT* "eavesdropping" on *RIGHT*'s UL transmission on SLOT 8, i.e., $SAMPLE_{LEFT}$ is:

```
D2D_CTRLMSG_RPT_MEAS received from Node 6 @ FID=41
D2D slot 8 direct-mode measurements:
 =>Rx SN: 228 229 230 231 232 233 234 235 236 237
 =>RSSI : 204 198 199 193 200 213 202 200 198 193
```

While the result of *RIGHT* "eavesdropping" on *LEFT*'s UL transmission on SLOT 7, $SAMPLE_{RIGHT}$ is:

```
D2D_CTRLMSG_RPT_MEAS received from Node 9 @ FID=34
D2D slot 7 direct-mode measurements:
 =>Rx SN:  99 100 101 102 103 104 105 106 107 108
 =>RSSI : 292 289 294 275 278 282 281 303 289 296
```

Even though both devices are transmitting packets over the air with the *same power*, the average RSSI values in the $LEFT \rightarrow RIGHT$ direction is approximately 280 compared to the average RSSI values in the opposite $RIGHT \rightarrow LEFT$ direction of around 200.

We attribute this phenomena to hardware imperfections in the "identical" WARP-boards we have, as well as to asymmetries in the direct radio channel between the two devices due to reflections, fading effects, etc.

In the algorithm design stage a decision could have been made to only enable mode-transition if the reported sample's *average* RSSI is above some threshold value, $\text{RSSI}_{threshold}$. If this had indeed been the case then different average RSSI values from the measurement samples $SAMPLE_{LEFT}$ and $SAMPLE_{RIGHT}$ above could have introduced complications in the mode-transition mechanism, with "D2D-mode" being enabled in one direction, but not the other.

The key factor here, made obvious by the prototyping activity, is that seemingly identical devices may still have differing characteristics, and these should be considered and taken into account when designing the algorithms and protocols for new concepts.

5. CONCLUSIONS

We set out with the goal of investigating a new wireless radio concept, namely, NA-D2D. Mechanisms needed to enable this feature were proposed, designed, and subsequently prototyped using our software radio wireless nodes.

We found that on the whole, the concepts we envisioned behaved as expected. However, we also discovered some subtle real-world issues which were not anticipated when initially designing these mechanisms. These include inherent delays in the system, as well as non-identical behaviour of similar hardware. Feeding these discoveries back into the design process resulted in a more robust system in the subsequent iteration of the prototype.

More generally, we foresee only positive benefits in utilizing small-scale software radio platforms for fast prototyping activity as part of a design process for new concepts.

6. ACKNOWLEDGMENTS

We would like to acknowledge the significant contributions by members of the EXCITE project at CalIT2 at UCSD with whom we have been collaborating with for many years. Their contributions, specifically in the development of the hardware design for the WARP-board, has enabled us to build this proof-of-concept implementation and further explore this new and exciting area of wireless communications.

7. REFERENCES

[1] 3GPP. Overview of 3GPP Release 12 V0.0.8. Description document, 3rd Generation Partnership Project (3GPP), 2013.

[2] K. Amiri, Y. Sun, P. Murphy, C. Hunter, J. R. Cavallaro, and A. Sabharwal. WARP, a Modular Testbed for Configurable Wireless Network Research at Rice. In *Proceedings of IEEE SWRIF*, 2007.

[3] M. Balleschi and G. Fodor. Performance analysis of Network-Assisted Device-to-Device communications. In *Swedish National Computer Networking Workshop (SNCNW)*. Linköping, Sweden, June 2011.

[4] M. Balleschi, G. Fodor, and A. Abrardo. Performance analysis of a distributed resource allocation scheme for Device-to-Device communications. In *IEEE Workshop on Machine-to-Machine Communications*. Houston, TX, USA, December 2011.

[5] P. Murphy, A. Sabharwal, and B. Aazhang. Design of WARP: A Flexible Wireless Open-Access Research Platform. In *Proceedings of EUSIPCO*, 2006.