A DTN Stack for Cognitive Radio Networks

Jawad Seddar, Hicham Khalifé, Vania Conan, Jérémie Leguay
Thales Communications & Security
Gennevilliers - France
name.surname@thalesgroup.com

Abstract— **In this paper, we evaluate over a software defined radio testbed the performance of a DTN implementation in cognitive radio networks. Our experimental platform uses USRP devices operated by the GNU radio platform for physical and MAC layers operations. Over the deployed software archietcture, we have compared 2 existing DTN implementations then run a set of expirements in order to chatacterize the reaction of the DTN bundle protocol to dynamically frequency changing environment. Our experiments show that DTN can offer lower delivery delays in cognitive radio networks. This improvement can reach 35% over highly dynamic spectrum bands.**

I. Introduction

The scarcity and high competition for wireless resources in some specific spectrum bands have triggered the emergence of the cognitive radio concept [1]. Enabled by the quick development of software defined radios, these smart and programmable radios have become the forefront of wireless communication research as they promise to offer reliable communications as well as efficient spectrum sharing. However, from an application point of view ensuring continuity in communications to end users when a channel switching occurs remains a challenge to overcome. This channel switching (sometimes called spectrum handoff) can be due to the arrival of a primary radio on the used band or even to the availability of a channel offering better performance.

To cope with the unstability of the wireless resources caused generally by interference and mobility, the Disruption and Delay Tolerant Networking (DTN) paradigm has been proposed [2]. The DTN concept was originally applied to *1)* static wireless networks suffering from frequent link disconnections, to *2)*Mobile Ad hoc NETworks (MANET) were connectivity issues are mainly related to nodes mobility as well as to *3)*inter-planetary networks and periodic message ferries where nodes mobility is known and efficiently exploited by the DTN mechanism. However, the cognitive radio concept that exploits opportunistically the wireless spectrum forces application to cope with disconnections caused by channel switching. Indeed, in the context of cognitive radio networks, in addition to the experienced interference on every channel, the spectrum mobility i.e channel switching can create interruptions in the ongoing communications.

In this paper, we apply the DTN concept to the cognitive radio networks. Through a real implementation over a software radio testbed we evaluate the gains offered by these technologies in dynamically changing spectrum conditions. To do so, we integrate a full software architecture shown in Figure 1 in order to enable DTN operations over software radio devices. We also compare in our testbed the performance of existing DTN implementations. Most importantly, we show in multihop topologies that DTN exploits opportunities created by intermittently available bands thus increasing throughput and reducing considerably delivery delays.

This work is supported by the EDA MIDNET project

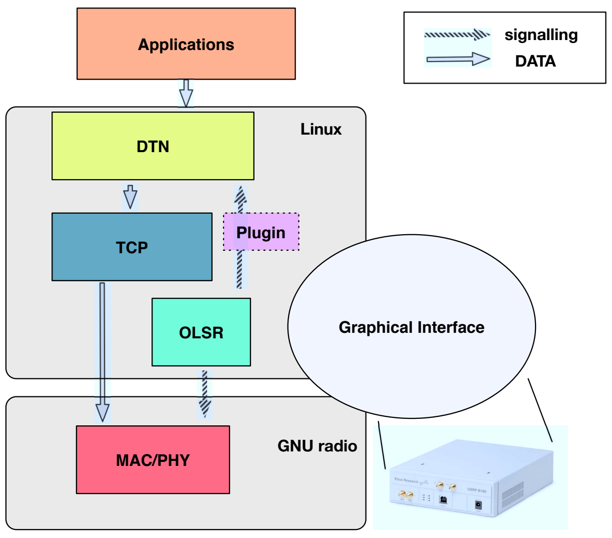


Fig. 1: Implemented software architecture

We argue that empowering secondary nodes in cognitive radio networks with DTN capabilities not only enhances the performance of these networks but also reduces the impact on the primary networks. Indeed, exploring data caching and custody transfer reduces the amount of exchanged information and hence moderates the undesirable effects of cognitive radio nodes on primary radios the latters having higher priorities on the shared resources.

The remainder of the paper is structured as follows: we first start by giving a brief overview on the DTN paradigm in Section II, then describe the software defined radio platform used in our tests in III. Section IV defines the implemented software architecture and the interaction between modules, experimentation results are given and discussed in V, finally SectionVI concludes the paper and gives future research directions.

II. DTN in a Nutshell

The DTN paradigm initially proposed by K. Fall [2] targeted intermittently connected networks. These connectivity interruptions can be mainly due to wireless links connectivity disruptions or to predictable and unpredictable nodes mobility. Since the first DTN paper a lot of research has been conducted in this area [3]. We give herein a short description of the key DTN mechanisms and motivate its use in cognitive radio networks.

The DTN concept emcompasses the required mechanisms to offer end-to-end connectivity in unreliable wireless networks. In order to achieve the promised goals, a DTN architecture is defined in rfc 4838 [4] with the following main novelties:

- *Bundles as data units* - The Bundle protocol is one of the most basic novelties introduced by the DTN architecture [5]. It can be seen as a new sublayer within the application layer of the protocol stack to constitute the data units in DTN networks. These data units are of arbitrary length. This protocol offers more flexibility in conveying data in unreliable networks by enabling partial data transfer between nodes thus data recovery after failures.
- *Store and forward* - Storing bundles even by unintended nodes constitutes a key factor in coping with link failures. Indeed, by buffering observed bundles the delivery success can be enhanced. This is partic ularly true when links appear and disappear oppor tunistically. Nevertheless, such mechanisms raise the challenge of dimensionning and managing th buffers added in each DTN node. Several studies an proposals to decide what to store, where and whe can be found today in the literature.
- *Addressing and late binding* - DTN adds a specifi Endpoint Identifier to identify DTN nodes. Further more, these DTN identifier are binded to the lower ad dressing scheme (IP addressing) in each node. Conse quently, DTN can exploit the already running routin protocols to detect new neighbors then decide for th most appropriate path to the destination.
- *Custody transfer* - Constitutes an important reliabil ity feature in lossy networks. In fact, this mechanism allows bundles to move closer to the destination b dropping the messages at the source whenever neigh bors acknowledge the reception of these messages. Such techniques allow intermediate nodes to retransmit bundles if connectivity opportunities occur. Practically, choosing the best custodian nodes on the path based on their available resources (caching, connectivity etc) and the efficiency and reliability of bundles delivery constitute a challenge to overcome.

III. Cognitive Radio Platform

We discuss herein the cognitive radio platform we use in our DTN stack implementation.

A. Software Defined Radio Devices

In order to implement our DTN architecture, we use the Universal Software Radio Peripherals (USRP) manufactured by Ettus Research [6]. In our tests, we rely on USRP2 devices (USRP N210), which are the last generation of the USRP products commercialized by Ettus.

The USRP2 is a radio device built around a FPGA. It possesses two 14 bit Analog-to-Digital Converters (ADCs) running at 100MSamples/s and two 16 bit Digital-to-Analog Converters (DACs) operating at 400MSamples/s. This enables us to have two complex channels simultaneously (2 I channels and 2 Q channels). Therefore, one complex input and one complex output can be simultaneously exploited.

This software defined radio is controlled through particular programs running on a computer (described later). The communication to the computer is done through a Gigabit Ethernet connection linking the computer directly to the FPGA. The USRP2 motherboard has two extension slots on which several kinds of daughter boards can be plugged. In our experimental setup, we use 2 types of dual-slot daughter boards:

- The WBX which operates between 50 and 2200 MHz thus covering the GSM 900 MHz band.
- The SBX which operates between 400 and 4400 MHz thus covering both the GSM 900 MHz band and the 2.4 GHz ISM band

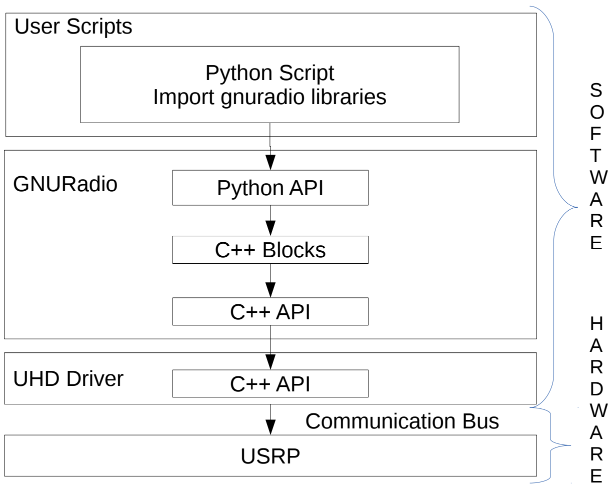


Fig. 2: USRP & GNU Radio software stack representation

B. GNU Radio

GNU Radio [7] is a free and open-source software development toolkit that provides signal processing blocks to implement software radios. It can be used with readily-available low-cost external RF hardware to create software-defined radios, or without hardware in a simulation-like environment.

GNU Radio applications are primarily written using the Python programming language, while the supplied performance-critical signal processing path is implemented in C++ using processor floating-point extensions, when available. Thus, the developer is able to implement real-time, high-throughput radio systems in a simple-to-use, rapid-application-development environment.

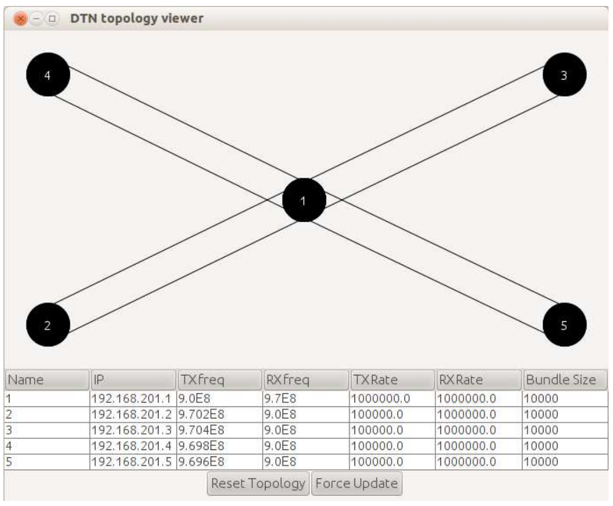
In order to control software defined radio devices, the GNU radio framework requires an additional convergence layer. For this particular reason, Ettus provides the USRP Hardware Driver (UHD) to link to their hardware. This UHD is provided as a standalone driver, and is made available to the GNU Radio toolkit through the implementation of several blocks, such as an emitter (uhd.usrp_sink), a receiver (uhd.usrp_source), etc.

Therefore, the overall system architecture can be thought of as a stack with the hardware (USRP device) sitting at the bottom of it. UHD is the direct link to the hardware and GNU Radio is the link between user defined flow graphs and UHD. Although one could directly connect to the hardware through UHD and without the use of GNU Radio, such configuration limits the functionning to simple operations while the GNU Radio toolkit is very furnished. The complete hierarchy is shown in Figure 2.

IV. Implemented Architecture over USRP platform

In order to enable DTN in a cognitive radio environment, we have implemented the architecture highlighted in Figure 1 over software radio devices. We have used the URSP N210 radios with the help of the GNU radio framework. Our architecture is composed of the following modules:

- *Applications* that we have developed in order to undergo message segmentation into tunable size DTN bundles.
- *DTN module* that offers buffering capability and topology management for the message exchange. Indeed, the DTN module transfers a whole message to the next hop and not the final destination by establishing a TCP connection between these two peers. We have compared in our experiments 2 DTN implementations DTN2 and IBR-DTN before selecting the most appropriate one to our context.
- *TCP protocol* that ensures the reliability and congestion control between communicating nodes. Note that the TCP protocol establishes one hop connection between the neighbors defined by the DTN module.
- *OLSR protocol* that discovers and establishes routes between deployed USRP devices in our test environment. Through a dedicated plugin we have developed, the OLSR module feeds the DTN module with topology information. We use the OLSR v0.6.6 release, and install it in the linux kernel
- *GNU radio framework* that implements the PHY and MAC layers over the SDR devices and described earlier in this paper.
- *Graphical User Interface (GUI)* that we have developed in Java. Our interface enables users to modify graphically the USRP device parameters (Frequency bands), then displays the topology as established by OLSR and observed by the DTN module. A screenshot of our GUI is displayed in Figure 3.



Name	IP	TXfreq	RXfreq	TXRate	RXRate	Bundle Size
1	192.168.201.1	9.0E8	9.7E8	1000000.0	1000000.0	10000
2	192.168.201.2	9.702E8	9.0E8	100000.0	1000000.0	10000
3	192.168.201.3	9.704E8	9.0E8	100000.0	1000000.0	10000
4	192.168.201.4	9.698E8	9.0E8	100000.0	1000000.0	10000
5	192.168.201.5	9.696E8	9.0E8	100000.0	1000000.0	10000

Fig. 3: Graphical User Interface showing a 5 node star topology

V. Experiments and Results

We describe in this section the conducted experiments over our softawre defined radio platform. We first compare 2 DTN implementations then highlight the advantange of using DTN over dynamically changing spectrum bands.

A. *DTN2 vs. IBR-DTN*

Several implementations of the bundle protocol exist. For this reason, the first step in our experimentation process focuses on comparing the most known open source DTN implementation. Our objective is to select the most appropriate version to our validation environment. We focus in our studies on the following two implementations:

- DTN2 [8], the reference implementation from DT-NRG (DTN Research Group in the IRTF)
- IBR-DTN [9], a portable implementation of the bundle protocol designed to run on embedded systems. It runs on standard linux distribution as well as several embedded systems and on Android smartphones.

We compare DTN2 (v2.9.0) to IBR-DTN (v0.10.2) using delay as our main metric. Over our experimental platform, we measure Round Trip Time (RTT) in different situations (locally over the same node, 1 hop, 2 hops) using the dtnping command provided by each of these implementations. The purpose of our tests is to observe at the DTN level the time required to send a message to the receiver and get back the answer. Clearly, this delay includes the time induced by TCP and lower layers (similar for both implementations). We also measured the delivery time of a 500 kB file over 1 hop. Results shown in Table I clearly demonstrate that IBR-DTN is much more performant in our setup than DTN2. It is also interesting to notice that IBR reduces the observed delays by a factor of 6 for ping messages and nearly 5 for the file exchange. This is probably due to the fact that IBR-DTN is optimized for embedded implementations.

TABLE I: Comparison between DTN2 and IBR-DTN

Application	DTN2	IBR-DTN
Average localhost ping (ms)	292	43.5
Average 1-hop ping (ms)	631.3	97.1
Average 2-hop ping (ms)	956.2	151.6
1-hop file transfer duration (s)	30.9	6.5

In light of these results, we have selected the IBR-DTN bundle protocol to use in all the following experiments of this paper.

B. *Impact of disruption*

After selecting the appropriate bundle protocol to operate over our software radios, we characterize the impact of using DTN in cognitive radio networks on throughput and delivery delays. We first consider the topology in Table 4 where the source tries to send a file to the destination by going through the relay node. We enforce the following disruption scenario :

- We start sending from node source to node destination.
- After 10 seconds, we start a disruption pattern on the link between relay and destination. The disruption occurs for a period of 1 second and repeats every 2 seconds.

This disruption pattern represents a highly unstable channel that can be used frequently by secondary users but for very short periods of time.

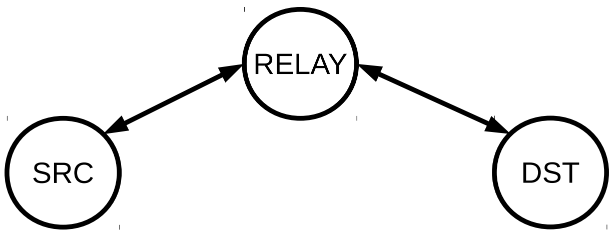


Fig. 4: Disruption topology

Figure 5 clearly shows that DTN outperforms TCP in terms of delivery time. This is due to the fact that in a DTN network, connections are hop by hop while TCP uses a source to destination connection. This forces TCP to retransmit from the source to the destination every time a disruption occurs. DTN on the other hand, only needs to retransmit from the relay to the destination as it buffers bundles in the relay node.

C. *DTN in a Dynamic Environment*

In order to evaluate the DTN resilience to topology changes due to spectrum mobility, we now consider the topology shown in Figure 6a. In this experiment, a source node (S) tries to send the same file to 2 destination nodes (D1 and D2). The scenario runs as follows:

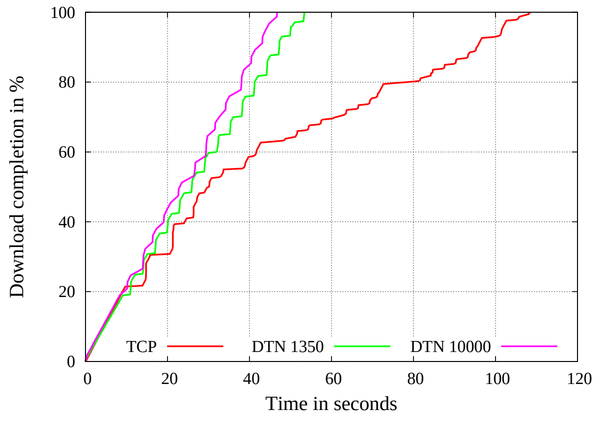


Fig. 5: Disruption results

- The source (S) starts sending the same message to both destinations at T_0.
- At T_1 seconds, the station (D2) is disconnected, as shown in Figure 6b.
- At time T_2 seconds, D2 reappears as a neighbour of D1. D1 serves now as a relay for D2. The resulting topology is shown in Figure 6c.

In practice, this scenario reproduces the impact of the arrival of a primary node on the selected channel between S and D2, thus forcing them to vacate. Moreover, the availability of a new band between D1 and D2 highlights the strict correlation between channel availability and the topology in cognitive radio networks.

In the results we show below, the S to D1 and S to D2 links have the same capacity of 100 $kbps$. The D1 to D2 link has a capacity of 1 $Mbps$. We investigate then the impact of the values of T_1 and T_2 on the delivery delay and throughput.

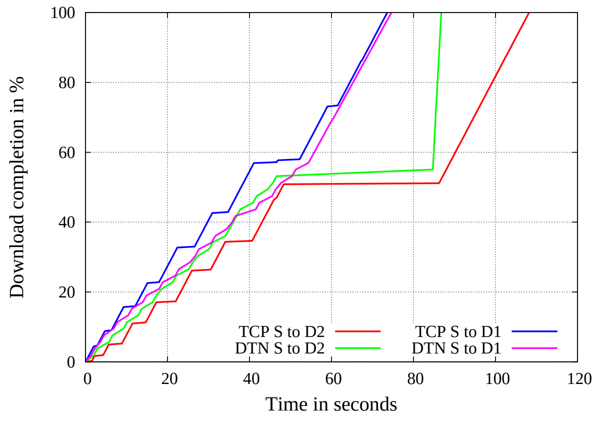
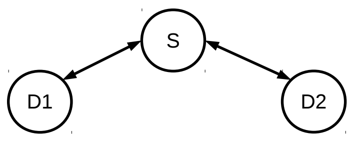
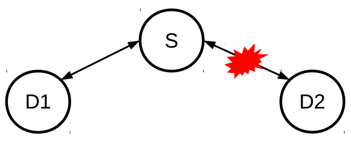


Fig. 7: Results for $T_1 = 50s$ and $T_2 = 80s$

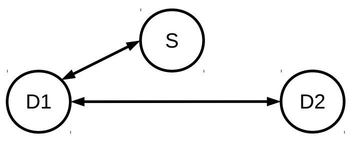
In Figure 7 we consider $T_1 = 50s$ and $T_2 = 80s$. Those values give enough time for S to complete the transfer to D1 while D2 is disconnected. Indeed, as shown in 7 the S to D1 tranfer takes around 60 seconds, while the link between D1 and D2 appears after 80 seconds. When D2 connects to D1, we clearly see that the DTN transfer is much



(a) Initial topology



(b) Topology after disconnection



(c) Topology after reconnection

Fig. 6: Evolution of the topology during experiment

faster than the TCP transfer. This is due to the fact that when using DTN, with the custody transfer functionality, D1 stores the bundles for D2. Therefore when D2 connects to D1 it can directly get the bundles from D1 instead of getting them from S. In TCP, D1 does not receive any packet for D2 until the new route is established (i.e going through D1). The connection has to continue from where it stopped from S to D2 with D1 as a relay.

If we focus on the case where D2 reconnects to D1 before the S to D1 transfer is finished, we observe the results shown in Figure 8. In such situations, DTN allows D2 to retrieve available bundles from D1 directly. This effect can be seen in the figure with the sharp slope of the S to D2 curve that starts at 50 seconds (T_2). Then after catching up with the buffered bundles, D2 receives the last bundles at nearly the same rate as D1 (plus the relaying time).

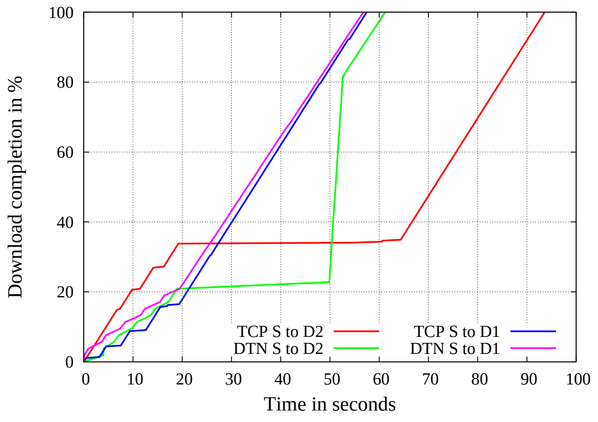


Fig. 8: Results for $T_1 = 20s$ and $T_2 = 40s$

Our experiments over a software defined radio platform show that exploiting the DTN concept can improve the delivery time by up to 35% (Figure 8). Therefore using advanced mechanisms enabled by DTN such as buffering, custody transfer can help overcome throughput limitations in dynamic cognitive radio networks.

VI. Conclusion and Future Work

We have presented here a software architecture that integrates the DTN concept over software radio devices. Our architecture exploits the GNU radio framework over which routing, transport and DTN protocols were implemented. The hence build software radio testbed allowed us to evaluate the assets of the DTN concept in cognitive radio environments. Our experiments lead to 2 important results. First, the IBR-DTN implementation is more efficient over the considered tesbed, second, the DTN, by using advanced mechanisms such as improved buffering and custody transfer, improves the communications performance in terms of delays and throughput.

In the future, we intend to test over our software architecture the performance of common applications such as VoIP and video streaming. Indeed, testing the users Quality of Experience (QoE) when the DTN is used in cognitive radio environment will allow to assess the usability of these concept in tomorrow's communication technologies. Moreover we plan to enrich our software defined radio platform with additional features and protocols.

References

[1] Akyildiz, Ian F. and Lee, Won-Yeol and Vuran, Mehmet C. and Mohanty, Shantidev, *NeXt generation/dynamic spectrum access/cognitive radio wireless networks: a survey*, Comput. Netw. Elsevier, 2006
[2] K. Fall, *A delay-tolerant network architecture for challenged internets*, Proc. ACM SIGCOMM, 2003
[3] M. Khabbaz and C. Assi, and W. Fawaz *Disruption-Tolerant Networking: A Comprehensive Survey on Recent Developments and Persisting Challenge* IEEE Communication Surveys and Tutorials, 2012
[4] IETF rfc 4838, 2007
[5] IRTF rfc 5050, 2007
[6] *Ettus Research* "http://www.ettus.com"
[7] *The GNU Radio project*, "http://gnuradio.org"
[8] *DTN Research Group* "http://www.dtnrg.org"
[9] *IBR-DTN* "http://trac.ibr.cs.tu-bs.de/project-cm-2012-ibrdtn"