Validation of a point to multipoint cognitive radio transport protocol over GNU radio testbed

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

***Abstract*—In this paper, we experiment on a GNU radio based platform a point to multipoint transport protocol specifically designed for cognitive radio environments. Over a dedicated software defined based platform deployed in our premises we investigate two main functionalities of transport protocols in terms of *reliability* and *congestion control*. Comparing with state of the art point to multipoint and even the TCP protocol as a benchmark, we show that our solution ensures high reliability and reacts judiciously to losses caused by interference. Moreover, our transport protocol is able to follow network conditions and channel switching by adapting its sending rate accordingly. Our extensive experiments also highlight that our solution is not only able to manage multiple destinations but also capable to offer better performance than TCP in terms of throughput over lossy links.**

I. Introduction

Recently, the cognitive radio paradigm arose as the forefront of wireless communication and networking research [1]. These research advances are supported by the emergence of off-the-shelf devices capable of exploiting multiple frequency bands and easily programmable by software, called software defined radios (SDR). In the meantime, new communication patterns in mobile computing are becoming very common with applications trying to take advantage of multi-interface devices and smartphones to distribute information between multiple users at the same time. These users may share socially a common interest or be located in geographically close positions. Disseminating alerts, videos and pictures to a set of workers or vehicles, in public safety networks or intelligent transport systems, are well known examples.

Few transport protocols were proposed with the objective of ensuring reliability and congestion control in point to point cognitive radio networks [2], [3], [4]. However, the sharing of content of a common interest to a group of users over various wireless links cannot be handled efficiently with these traditional transport protocols. In fact, challenges are related to sending at a suitable rate for all receivers, reacting to losses at particular destinations, avoiding that slow receivers penalize those benefiting from better links to name a few. In practice, this communication model addresses applications' need to distribute content, such as video streams, to multiple receivers sharing the same interests. Because in a point-to-multipoint context, users can be spread over different frequencies, channels, or locations, the links "connecting" each destination to the source, might have inherently different characteristics (e.g. bandwidth, center frequency and thus propagation properties, etc) or be interfered by different (primary or secondary) users (in a cognitive radio context). As a result, users in the same group may experience very heterogeneous performance in terms of latency, physical transmission rate, MAC layer retransmissions, etc.

To support point-to-multipoint distribution of the same data across heterogeneous receivers, the source can adapt its flow to the slowest receivers, like in the NORM [5] protocol or in RTMP [6]. This however translates in pulling down the reception rate of all nodes; while this might be appropriate in the Internet where receivers have close behaviors, it is not the case in a multi-channel wireless context. Alternatively the source could follow the fastest receivers, but it is taking the risk of "losing" the slowest, resulting in too many packets being dropped on the saturated slow channels. To solve this dilemma, we have proposed PMT (Point-to-Multipoint Transport) in [7], an acknowledgement based transport protocol which dynamically differentiates among receivers and separates them according to their reception capabilities. PMT creates dynamic groups of receivers managed by the source to improve delivery time for the nodes that can receive data early, and thus the overall throughput.

In this paper, we compare over an experimental cognitive radio platform based on USRP GNU radios [8] the proposed PMT protocol to known transport protocols. Our objective is to highlight the capability of our solutions to ensure reliability and efficient congestion and flow control in unstable and dynamically varying wireless environments. To do so, we contrast over a real platform PMT with state of art point to multipoint and even point to point (TCP) transport solutions. Indeed, even through TCP is not suitable for multi-destination communications it is used as a benchmark for the high reliability it offers. Our experiments show that as PMT guarantees high reliability for all destinations of a communication, it also adapts it flow control to the selected channels outperforming offering higher throughput than existing solutions.

The remainder of the paper is structured as follows. Section II. details the protocol and its mechanisms. Section III. describes our cognitive radio platform. We define and analyze our experimentation in IV emphasizing on the comparison with the TCP protocol. Finally, related work is given in V then conclusion and future work in Section VI.

This work is supported by the French National Research Agency (ANR) under grant reference LICoRNe ANR-10-VERS-005 and EDA MIDNET.

II. TRANSPORT PROTOCOL

A. Preliminaries

In a multidestination configuration, the throughput of a group composed of M members can be expressed as the sum of the throughput of all the members of the group.

$$\Phi_{group} = \sum_{i=0}^{M} \Phi_i$$

where Φ_i is the throughput observed by the member i of the group.

In order to prevent slow receivers from penalizing those benefiting from favorable network conditions, we seek to create dynamically separate groups each served at a particular throughput. We base the group formation algorithm on the round trip time (RTT) observed by each node. With these observations, the source node is able to differentiate between slow and fast nodes. A single time threshold T is sufficient to discriminate between both groups: all nodes below the threshold T go in the fast group, all nodes above in the slow group (their RTT is larger). In the considered setting, the trade-off is the following: one could try to maximize the number of receivers in the first group. However, this would imply increasing T and thus reducing throughput for all nodes in the first group. It is intuitive that an optimal value of T should exist, depending on the RTT probability distribution. Please refer to [7] for a detailed theoretical analysis.

B. Protocol description

Our mechanism is source driven, in other words the source node maintains, in a special database, the group affiliation for every receiver. The average RTT for every receiver (other participants within the zone) is also stored inside this database. Moreover two transmission buffers are added, each handling transmissions for a precise group. The protocol building blocks are shown in Figure 1.

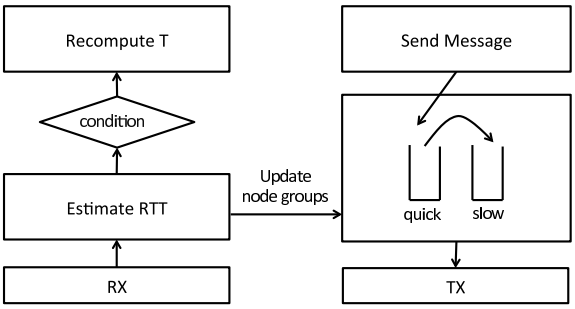


Fig. 1: Protocol building blocks

The source sends a message for the fast group every T and serves the slow receivers every T_{max}. In fact, the source transmits the message available in the fast nodes queue to the fast receivers and waits for the acknowledgements. After T seconds (i.e at the expiry of the fast nodes interval), receivers that have answered are labeled as fast; all others are labeled as slow in the specific database. Using timestamps, the smoothed RTT of fast receivers is also updated. The message is then transfered to the slow group buffer and transmitted to the slow receivers at T_{max}. Note that T_{max} can have an important role in tuning the rate and reliability of the PMT protocol. We have investigated 2 possible solutions. The first one considers T_{max} as a fixed protocol parameter, dictated by the application requirements, or the need to ensure quite slow nodes can still get enough packets. The second solutions optimizes T_{max} similarly to T in a way to maximize the slow group throughput. We have used this second option in our testbed experiments.

This flow control process is repeated whenever new messages are available for transmission. More generally, the throughput of our protocol is dictated by T and T_{max} as follows:

- at T source pushes the message to the slow group queue, pops a new message and sends it to the fast group receivers.
- at T_{max} source removes from the slow queue the message sent T_{max} seconds earlier, then transmits the message in head of queue to the slow group members.

Nodes can move from one group to the other depending on their RTT. From the fast to the slow group, nodes will ignore the messages that they have already acknowledged. From the slow to the fast group, the source will consider nodes in both groups so that they keep receiving old and new messages.

Since the objective is to handle dynamically rate vs. range requirements enabled by cognitive radios, the strategy of the protocol is to improve delivery time for the nodes that can receive data early. The long-term throughput of the system is unchanged, as it is dictated by the second queue (the slow nodes that are served every T_{max}), since all nodes receive the same data.

C. Algorithm for dynamic group calculation

In order to select the appropriate value of T that separates the fast from slow nodes, we propose a greedy algorithm which maximizes the average throughput per node (which is the same as maximizing total network throughput for a fixed number of receivers).

The basic idea of the algorithm is to determine the value of T by computing the average throughput based on the receivers' RTTs. First, we start by sorting received RTTs in increasing order (line 1). Then, by sequentially selecting the RTT of receiver j and computing the throughput of each group accordingly (i.e by also including all receivers having smaller RTT) we estimate the throughput as if the RTT of receiver j equals the value of T (line 3 of the algorithm). At the end of this loop the algorithm returns the RTT value that offers the highest total throughput. In practice, a slightly bigger (+ ϵ) value from this RTT is selected for T in order to maximize the total network throughput. In fact, this small margin allows to account for potential RTT fluctuations.

PMT flow control transmits a window of N segments to ever destination every T. These N segments generate a single

Algorithm 1 Estimate optimal value of T

Input: N //total number of receivers
$\tau[N]$ //table containing smoothed RTT of every receiver
T_{max}
$max = 0$, $index$, $result$, j //intermediate variables
Output: T

```
sort(τ[N])
while j < N do
    result ← (j.1/τ[j]) + (N−j).1/T_max // T_max = τ[N−1] + ε
    if result > max then
        max ← result
        index ← j
    end if
    j = j + 1
end while
T ← τ[index] + ε
return T
```

ACK from every destination that allow to detect and restore lost segments in the window. Indeed, reducing the number of ACK and increasing N the window size sent every T allow to maximize the network throughput.

III. Cognitive Radio Platform

We describe herein the cognitive radio platform we use to evaluate the performance of the PMT protocol.

A. Software Defined Radio Devices

In order to implement the previously detailed protocol, we use the Universal Software Radio Peripherals (USRP) made by Ettus Research [8]. In our tests, we rely on USRP1 devices, which are the first generation of the USRP products commercialized by ettus.

The USRP1 is a radio device built around a FPGA. It possesses four 12 bit Analog-to-Digital Converters (ADCs) running at 64 MSamples/s and four 14 bit Digital-to-Analog Converters (DACs) operating at 128 MSamples/s. This enables us to have four complex channels simultaneously (4 I channels and 4 Q channels). Therefore, up to two complex inputs and two complex outputs can be simultaneously exploited.

This software defined radio is controlled through particular softwares running on a computer (described later). The communication to the computer is done through a USB 2.0 connection linking the computer directly to the FPGA through a Cypress FX2 USB controller. The USRP1 motherboard has four extension slots on which several kinds of daughter boards can be plugged. In our experiment setup, we use 2 daughter boards of 2 slots each in order to fill the all 4 available extension slots

- The RFX900 which enables us to transmit and receive around 900 MHz (GSM frequency)
- The RFX2400 which enables us to operate around 2.4 GHz (ISM band)

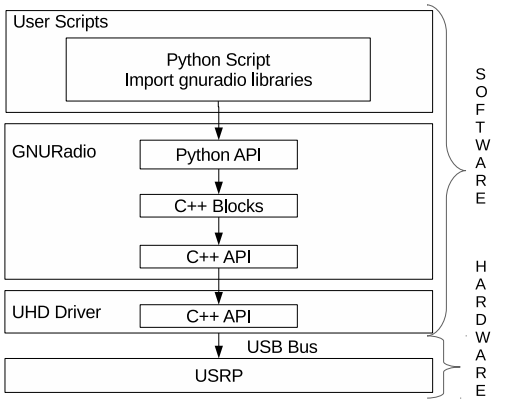


Fig. 2: USRP & GNU Radio software stack representation

B. GNU Radio

GNU Radio [9] 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.

GNU Radio is therefore a very useful software tool however it still needs an additional layer in order to control our software defined radio devices (III-A).

This is the role of the USRP Hardware Driver (UHD) provided by Ettus Research. It 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, they would be limited to simple operations while the GNU Radio toolkit is very furnished. The complete hierarchy is shown in Figure 2.

IV. Experimental comparison of transport protocols

Over this GNU radio based platform we compare our transport protocol (PMT) to existing solutions.To the best of our knowledge, experiments of cognitive radio protocols in general and particularly transport protocols over a real testbed are rarely conducted. We believe that undergoing extensive

experimentation and comparison of different protocols in a cognitive radio platform is a major contribution of our paper.

A. Considered Setup

The experimental validation of the PMT protocol requires at least 3 nodes: 1 emitter and 2 receivers. Our objective is to establish point to multipoint communication towards 2 destinations with 2 channels having different characteristics in terms of capacity and packet error rate. More precisely, we consider a high speed and reliable channel (less than 1% packet error rate, PER) offering 1 Mbits/s capacity and a lossy channel of a PER of 20% with a capacity of 250 Kbits/s. However, in order to deploy this setup over our cognitive radio platform, the source node is emulated by two GNU radio devices each transmitting exclusively over a frequency band. This was done to overcome the low physical capability of these experimental devices when a switching from a channel to another is required. Such configuration allows us to simply compare point to multipoint protocols to traditional point to point solutions by running experiments in one of the following setups (explained in Figure 3):

- The two USRPs are programmed as simple interfaces operating on different channels. On top of them, a single point to multipoint transport protocol is instantiated. This corresponds to a single node equipped with 2 interfaces.
- The two USRPs are managed as 2 independent nodes each serving a separate TCP source on a separate channel. Therefore, 2 TCP connections can be initiated in the Linux kernel and tunneled towards the radio devices using the socket API.

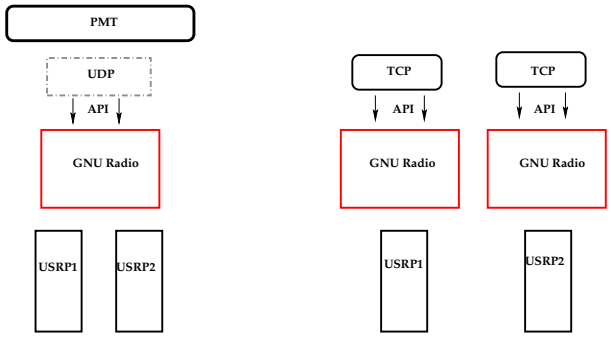


Fig. 3: PMT vs TCP implementation over USRP GNU radio devices

Throughout our experiments, we use the 2 configurations shown in Figure 3 to compare PMT reliability and congestion/flow control to TCP when exchanging 1 MegaBytes of data between the sender and 2 receivers. PMT window size was set to $N = 5$ segments. This value was statically selected since it offers a good trade-off between following closely network conditions and reducing the overhead caused by acknowledgments. Indeed, as long as link starvation is avoided i.e. the source finishes sending before the destinations start receiving any sent segment, any value of N can be used. Besides, a minimal MAC layer to regulate access control through a CSMA-like mechanism as well as a physical layer that ensures FEC through redundancy are implemented over our GNU radio devices.

B. General behavior

In the aforementioned configuration, we seek to investigate our solution behavior in different network conditions and highlight its ability to cope simultaneously with receivers observing links with different PER and capacities. Indeed, the comparison with TCP is justified by the following arguments:

- Evaluate the reliability and the delivery time of the data relatively to TCP, which is today the reference and the benchmark of transport solutions
- Highlight the ability of PMT to ensure transport services for multiple destinations with a single instantiation of the protocol. Unlike TCP, that requires a separate TCP connection per destination. (refer to Figure 3)

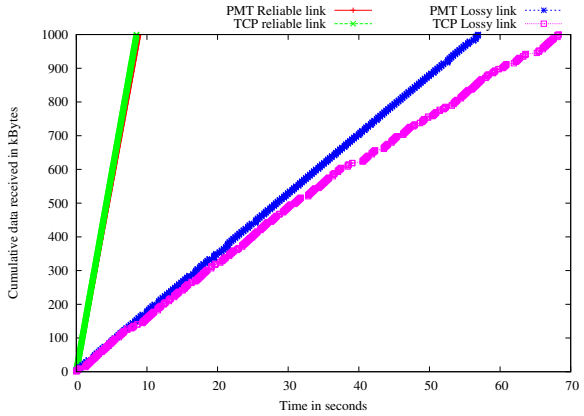


Fig. 4: PMT vs TCP comparison on two separate links

Figure 4 shows the amount of sent data in time for PMT and TCP Reno flavor at each receiver over the reliable and unreliable channels. The reliability mechanism of PMT offers similar performance to TCP by ensuring the delivery of the entire 1 MegaBytes message without any loss on both channels as shown in Figure 4. More interestingly, a single instantiation of our protocol is able to achieve similar throughput to TCP on a high capacity link and in the same time outperforms the TCP connection on the lossy link. This is due to the flow control mechanism employed by PMT that adapts the source sending rate to the measured round trip time. Moreover sending a single ACK for every 5 segments further increases the throughput of PMT. Consequently, over unreliable links, the TCP congestion control reduces its sending rate when losses are observed hence increasing the total duration of the message exchange. Whereas with PMT, the source reduces its sending rate (increase T) when the RTT become larger which constitute a more accurate estimation of congestion thus more suitable reactions. This is further discussed in the following section. Note that we have used Reno the default TCP flavor implemented in our Linux kernel. While other flavors of TCP include RTT estimation in their congestion control

mechanisms, even if these flavors can perform better in noisy environments, none of them is able to handle simultaneously multiple receivers and manage them dynamically.

C. PMT vs. TCP on lossy links

In order to compare the congestion/flow control of the two protocol under lossy conditions we study the evolution of TCP congestion window as well as the measured round trip time that dictates the source sending rate with PMT. In our experiments, TCP congestion window size $cwnd$ as well as slow start congestion threshold ss_{thresh} are directly obtained from the TCP variables maintained by the Linux kernel. For the PMT, we show the scheduled sending time for the slow group T_{max} and the measured RTT for this destination. It is worth mentioning here that since the lossy channel suffers also from low capacity (250 Kbits/s), the destination reached over this

channel falls in the slow group with PMT.

Platform experiment results shown in Figure 5 highlight the two protocols reactions to packet loss over the wireless medium. Practically, the lossy environment in our context corresponds to wireless links between the 2 GNU radio devices without FEC coding on the physical layer. The hence packet error rate we measure over this link is around 20%. The results of Figure 5 illustrate why the PMT outperforms TCP in terms of achieved throughput over lossy links. In fact, the congestion control mechanism of TCP reduces the sending rate (congestion window) when packet losses are observed. Moreover, if lost TCP segments are not retransmitted in time (i.e upon RTO expiry), the TCP protocol enters slow start phase thus reducing its congestion window to its lowest possible value. This effect can be seen in Figure 5a when the $cwnd$ value goes below the slow start threshold (ss_{thresh}) value. Clearly this happens very often in our experimentation. In contrast, PMT exploits exclusively the RTT measurement in computing the sending rate at the source. Occurring losses on the wireless links are retransmitted by the source however without impacting the RTT measurements. As a direct result, our protocol is able to closely follow the network conditions and adapt its sending rate accordingly as shown in Figure 5b.

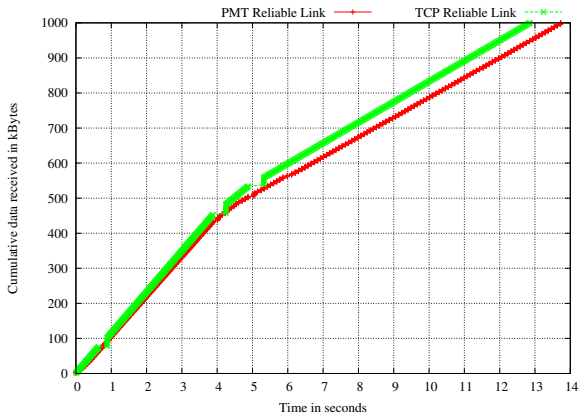


Fig. 6: PMT vs TCP over congested and unstable links

D. PMT vs. TCP reaction to congestion

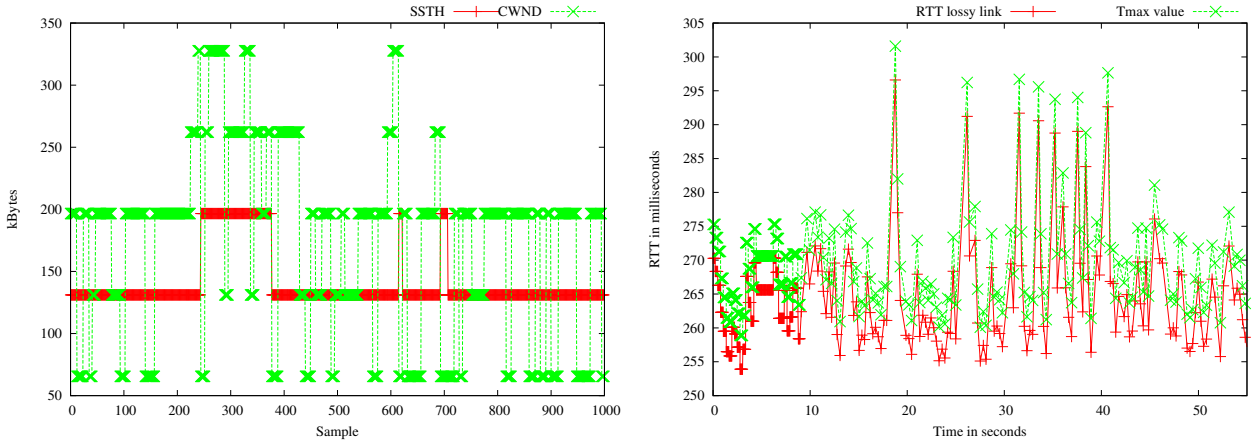
Because PMT is conceived to operate in point to multipoint cognitive radio networks, we investigate its reaction to congestion and link properties variation in Figure 6. To do so, we modify the link capacity over the reliable link initially at 1 Mbits/s to 700 Kbits/s at second 4 then to 500 Kbits/s at second 5. These capacity changes can be seen as a congestion through the start of a competing traffic or a channel switch in a cognitive radio environment that requires a dynamic adaptation of the transport protocol. Our experiments show that both protocols dynamically adapt to the network conditions by modifying the source sending rate. This can be observed in the change of the slope of the number of sent octets in time. Moreover PMT, by following the measured RTT adapts smoothly it sending rate unlike TCP that goes to the slow start phase when packets are lost and not retransmitted in time (refer to seconds 4 and 5 of TCP curve in Figure 6). Note here, that the additional delay for PMT to transmit the 1 MBytes message is induced by our demonstration setup that implements our protocol as an application and compares it to TCP directly connected to socket API. This configuration adds few milliseconds at each transmission for PMT what makes the 2 curves diverge at the end of the experiment. Note also that this effect is present in Figure 4 but not visible due to scale effects.

We have also compared PMT with the NORM protocol that creates a group of receivers with a sending rate adapted to the slowest receiver. Unsurprisingly, these real testbed measurements not shown here for space constraints show that PMT, by creating multiple groups, outperforms NORM in terms of achieved throughput.

V. Related Work

Recently proposed transport protocols for point to point cognitive radio networks [2], [4] do not address challenges of the point-to-multipoint communication scheme. In fact, these solutions make the comprehensive assumption that at time t a single destination needs to be reached. Hence rate adaptation is based on optimizing the transmission parameters based on this destination reception capabilities. Alternatively, present point-to-multipoint transport solutions do not cope well with the new conditions created by cognitive radio environments. Standard multicast solutions target essentially multicast sessions with large groups [10], [6]. For receivers with different flow rates, one can compute multicast groups based on throughput [11] or create layered multicast protocols [12]. They apply well to layered content/stream distribution, where each quality layer is mapped to the corresponding receiver rate.

In the meantime an increasing interest for the GNU Radio development kit have been observed in the wireless research community. These activities mostly focused on implementing SDR oriented solutions using GNU Radio in order to validate theoretical studies. Clearly, validating technical contributions in realistic environments hence going beyond simulations is highly encouraged in wireless communication and networking communities. This tendency is gaining momentum with the



(a) TCP congestion window

(b) PMT rate adaptation

Fig. 5: TCP vs. PMT control over lossy links

proliferation of low cost and highly efficient off the shelf configurable devices. However, most of the work in the GNU Radio community focuses on physical layer [13] [14] and MAC layer aspects [15], [16]. Very few efforts have looked on implementing higher layer solutions and protocols. In a sense, we are also using GNU Radio as a mean to implement our protocol and test its behavior in real-life conditions, nevertheless our approach, using GNU Radio in order to implement a point to multipoint transport protocol, is to the best of our knowledge one of the first initiatives towards this direction.

VI. Conclusion and Future Work

In this paper we have validated over a real cognitive radio platform an acknowledgement based transport protocol for point-to-multipoint multi-channel networks which splits receivers into groups, each served at a suitable throughput. The major advantage of our solution consists in preventing slow receivers from affecting the service offered to receivers possessing better conditions (fast group). Our experiments demonstrate first that TCP performs poorly in lossy environments. This observation is well accepted today in the community.

More importantly, these testbed experimentations highlight the capability of our transport solution (PMT) to ensure with a single instantiation of the protocol high reliability and efficient flow and congestion control, two main services of transport protocols. More precisely, PMT offers similar performance to TCP over reliable and high speed wireless links and outperforms TCP congestion control in lossy environments.

In the future, we plan to extend the solution to N groups. Intuitively, this can be seen as running the same algorithm recursively on the created groups. However, optimality of this solution should be verified in terms of obtained throughput for every group. On our platform, we envisage to test our transport protocol in multihop context where new problems related to hidden node and channel allocation arise and may impact its performance. Investigating the fairness of our solution with multiple flows competing for the same wireless resource is also envisaged in our future work.

References

[1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey," *Comput. Netw.*, vol. 50, no. 13, 2006.

[2] K. Chowdhury, M. D. Felice, and I. Akyildiz, "Tp-crahn: A transport protocol for cognitive radio ad-hoc networks," in *IEEE infocom conf*, 2009.

[3] D. Sarkar and H. Narayan, "Transport layer protocols for cognitive networks," in *IEEE infocom workshops*, 2010.

[4] C. Luo, F. R. Yu, H. Ji, and V. C. M. Leung, "Optimal channel access for tcp performance improvement in cognitive radio networks," *Springer Wireless Networks*, vol. 17, no. 2, 2011.

[5] B.Adamson, C.Bormann, M. Handley, and M. Handley, "Nack-oriented reliable multicast (norm) transport protocol," *RFC 5740.*, 2009.

[6] S. Paul, K. K. Sabnani, J. C.-H. Lin, and S. Bhattacharyya, "Reliable multicast transport protocol (rmtp)," *IEEE Journal of Seleted Areas in Communication (JSAC)*, vol. 15, no. 3, 1997.

[7] H. Khalife, V. Conan, J. Leguay, and T. Spyropoulos, "Point to multi-point transport in multichannel wireless environments," in *IEEE WCNC conf*, 2013.

[8] "Ettus Research." [Online]. Available: http://www.ettus.com

[9] "The GNU Radio project." [Online]. Available: htpp://gnuradio.org

[10] G.Kwon and J.Byers, "roma: a reliable overlay multicast with loosely coupled tcp connections," in *IEEE infocom conf*, 2004.

[11] L.Rizzo, "pgmcc: a tcp-friendly single-rate multicast congestion control scheme," in *ACM SIGCOMM conf*, 2000.

[12] G.Kwon and J.Byers, "Smooth multirate multicast congestion control," in *IEEE infocom conf*, 2003.

[13] M. Braun, M. Mller, M. Fuhr, F. K. Jondral, "A USRP-based Testbed for OFDM-based Radar and Communication Systems," *Proc. of 22nd VTech Symposium on Wireless Communications*, 2012.

[14] J. Zhang, J. Jia, Q. Zhang, and E. Lo, "Implementation and evaluation of cooperative communication schemes in software defined radio testbed," in *IEEE infocom conf*, 2003.

[15] A. M.-T. A. Puschmann, M. A. Kalil, "A Flexible CSMA based MAC Protocol for Software Defined Radios," *Frequenz.*, vol. 6, no. 9–10, 2012.

[16] G. Nychis, T. Hottelier, Z. Yang, S. Seshan, P. Steenkiste, "Enabling MAC Protocol Implementations on Software-Defined Radios." *NSDI'09 Proc. of the 6th USENIX*, 2012.