TFRC-CR: An Equation-based Transport Protocol for Cognitive Radio Networks

Abdulla K. Al-Ali and Kaushik Chowdhury
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115
Email: {al-ali.a, krc}@husky.neu.edu

***Abstract*—Data delivery in a dynamically changing spectrum environment continues to remain an unsolved problem, with existing TCP based implementations falling short owing to their inability to react swiftly to spectrum changes. This paper proposes the first equation-based transport protocol, based on the TCP Friendly Rate Control (TFRC) protocol, which uses the recent FCC mandated spectrum database information instead of relying on any intermediate node feedback. Not only does this approach maintain the strict end to end property required at this layer of the protocol stack, but also allows fine adjustment of the transmission rate through continuous adaptation. We explore interesting directions on how to limit repeated queries to the spectrum database and yet allow the source to control the rate effectively; when to re-start the transmissions; and how to interpret possible spectrum changes in the intermediate nodes correctly without mistaking it for normal network congestion, among others. Our extension to the ns-2 simulator enables thorough testing of various aspects of our protocol adapted for cognitive radio, called as TFRC-CR. We show through simulation an improvement of over 33% in the end to end throughput when compared with the classical TFRC.**

I. Introduction

Cognitive radio (CR) networks enable opportunistic use of available licensed spectrum, and reduce the pressure on the unlicensed ISM bands in the $2.4\,\mathrm{GHz}$ and $5\,\mathrm{GHz}$ range. While the core functions of spectrum sensing, switching, and sharing are now being better understood given the rapid strides in this area, work on higher layers of the protocol stack, such as the transport layer, is still in a nascent stage.

There has been extensive investigation of window-based TCP protocols for assuring congestion-free behavior in classical wireless networks, given its widespread use in the wired domain. By minor modifications of the information contained in the feedback acknowledgments (ACKs) sent by the destination, such as by falsely advertising a receive window of 0 in Freeze TCP [6] when an impending handoff is detected, the TCP source can be prevented from transmitting. The single end-to-end connection can be split into the wired (sender to base station or BS, when such an infrastructure support exists) and wireless (BS to the wireless node) planes, as shown in WTCP [7]. In Addition, some protocols explore tuning the sender's transmission rate through explicit notifications (TCP EFLN) [8] and via selective retransmissions of lost packets (TCP SACK) [9]. While each of these approaches have merits, they were not originally designed with the aim of licensed or primary user (PU) protection, sudden large-scale bandwidth fluctuations, and periodic interruptions caused by spectrum sensing and channel switching. A recent approach called TP-CRAHN addressed these concerns [1], though it relied on extensive feedback from the underlying layers of the protocol stack as well as the intermediate nodes that form the connected chain. This is undesirable as it violates the end-to-end paradigm typically assumed for transport layer protocols.

This paper is aimed to open up fresh discussion on the design of CR-specific protocol using an equation-based approach, wherein the concept of the congestion window in classical TCP (and the self-clocking via returning ACKs) is no longer followed. A notable example in this area for wireless ad hoc networks is ATP [4], which argues that the short term unfairness of the CSMA/CA MAC results in an undesirable bursty data flow. This problem is exacerbated in CR networks because nodes pause the transmission when they are engaged in sensing or channel switching. This, in turn, results in varying round-trip time estimates (in the case of TCP) and higher induced load rendering the self-clocking nature ineffective. The frequency and reliance on the ACKs for window based transmissions also lead to reverse path performance impact on the forward DATA path. In TCP, this can amount to 10%-20% of the data stream rate [4]. The bursty nature of TCP is clearly demonestrated in Figure 1. In this paper, we devise the first equation based transport protocol using existing approach called as TCP Friendly Rate Control (TFRC) [2]. We name our modified design as TFRC-CR, whose main features are as follows:

- It allows the TCP source to integrate with a designated spectrum databases, as mandated by the FCC in a recent ruling [5]. There is no feedback from any of the intermediate nodes, and neither is any information from an underlying layers is utilized.
- It enhances the speed of response by distinguishing between the effect of spectrum change and true congestion, by leveraging both the information from the spectrum database and the periodic information sent by the destination through the usual ACKs. Hence, the transmission rate is almost never penalized unless the need is justified and likewise, the coarse jump in the rate can be much faster than that possible in traditional transport protocols.

978-1-4673-0009-4/13/$26.00 ©2013 IEEE

TABLE I: Symbols used by TFRC-CR's framework

Symbol	Description
s	Packet size
RTT_{avg}	Average round-trip-time
RTT_{stddev}	Round trip time standard deviation
I_n	Average ACK inter-arrival during no PU activity
I_{PU}	Average ACK inter-arrival during PU activity
t	Time now
t_n	Time last ACK received during no PU activity
t_{PU}	Time last ACK received during last PU activity

- It intelligently polls the spectrum database only when needed, by identifying a possible PU arrival event, i.e., it does not consume the backend system resources used for interacting with the database. The database may likely be a single point of failure and hence, it needs to be protected from being overwhelmed in a practical situations. The current regulations require database polling at least in 60 second-intervals, and our aim is to exceed that default value only when a critical need is detected.

The rest of this paper is organized as follows: The preliminary background of TFRC and the motivation for adapting it for CRs is described in Section II. In Section III, we describe the proposed protocol TFRC-CR in detail. Section IV gives results from our comprehensive simulation study, and finally, we conclude our paper in Section V with pointers to future research.

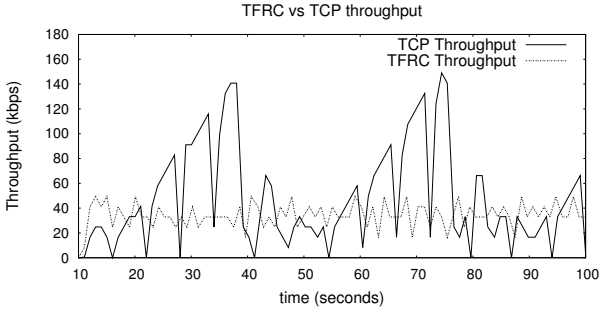


Fig. 1: Throughput comparison between TCP and TFRC in a 3-hop chain ad-hoc network

II. TFRC Background and Motivation

TFRC is a rate based mechanism for congestion control in unicast traffic. We use this as the platform to build our protocol because it aims at providing a stable throughput, as opposed to the sudden fluctuations caused by Additive Increase Multiplicative Decrease (AIMD) in TCP, while maintaining a TCP friendly rate. Given that TFRC is also an end-to-end protocol, deployment is only necessary at the source and destination. To achieve reduced fluctuations, it calculates a parameter called the *loss event rate* (p) at the receiver by taking the reciprocal of the average weights in the last (n) loss event samples where a sample is defined as the number of consecutive data packets received within a RTT before a packet loss happens. TFRC, by default, averages the last 8 samples ($n = 8$) which gives an approximate snapshot of how the traffic flow looked in the last n RTTs. p is then sent to the sender to be used in the throughput equation that estimates TCP's average sending rate:

$$X_{bps} = \frac{s}{R * \sqrt{\frac{2*b*p}{3}} + (t_{RTO} * (3 * \sqrt{\frac{3*b*p}{8}} * p * (1 + 32 * p^2)),} \quad (1)$$

where X_{bps} is TCP's average transmit rate in bytes per second, s is the packet size in bytes, R is the round-trip time in seconds, p is the loss event rate between 0 and 1.0, t_{RTO} is TCP's retransmission timeout value in seconds, and b is the maximum number of packets acknowledged by a single TCP ACK.

The goal of this is to achieve a smoother rate by avoiding sudden abrupt rate fluctuations as is the case in TCP.

TFRC suffers when deployed in Cognitive Radios owing to the following reasons: (i) *Low rate after PU exit*: TFRC is not able to use the maximum allocated bandwidth after a PU exits, due to TFRC weighing the last loss rates which are recorded falsely during PU activity, ii) *Slow recovery*: After the PU exits, having mistaken the original interruption purely as a congestion event, TFRC starts polling the bandwidth connection over large intervals of time, which can lead to significant delays in resuming the transmission, and iii) *Buffer overload and interference*: TFRC initially sends multiple packets during the PU activity as part of its regular rate control, causing additional interference with the PU. These individual issues are described in further detail in the rest of this section.

• *Low rate after PU exit*: After a prolonged idle state due to PU activity, the last n loss event rates should be neglected because they occurred in that stagnant state. This causes TFRC to resume the throughput at a false loss event rate p leading to less than optimal throughput. TFRC will recover after at least n event loss rates have been recorded. This can take up to n Round-Trip-Times (RTT).

• *Slow recovery*: During PU Activity, TFRC's $nofeedback$ timer expires several times which leads to reduction in the effective rate by half each time until the minimum rate of $\frac{s}{t\_mbi}$ where s is the packet size and t_mbi is set to 64 seconds is reached. This means that TFRC will poll the network once every 64 seconds which can cause a delay of up to 64 seconds for the network to resume activity after a PU exit.

• *Buffer overload and interference*: During the PU activity, TFRC will initially continue the ongoing rate because it is unaware of the PU activity. This will lead to undesired packets being sent over the network leading to a bigger buffer queue at the MAC layer of the node immediately before the affected area. These packets will retransmit until they timeout which causes additional undesired interference with the PU.

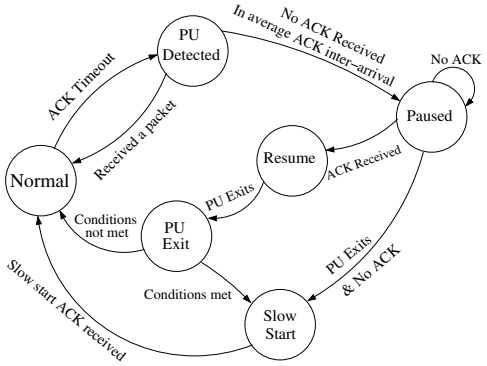


Fig. 2: TFRC-CR finite state machine

III. TFRC-CR: An equation based Transport Protocol

In this section, we describe our proposed TFRC-CR protocol and the modifications done to classical TFRC. Our changes in the protocol operation are aimed at (i) ensuring that the connection resumes the data stream immediately after the PU leaves the affected traffic zone, (ii) striking a balance between polling the network too frequently by the source that may cause interference with the PU, and conversely, reacting too slowly to spectrum change, and (iii) finding the available bandwidth as soon as possible, after the PU vacates the spectrum.

We present an overview of our approach using a finite state machine diagram, as shown in Figure 2. The remainder of this paper refers to the symbols listed in Table I.

A. Normal state

This is the default state of TFRC-CR, and the protocol returns to this state whenever there is no expected spectrum outage. The response during true congestion events and the resulting change is the transmission rate is identical to that of the classical TFRC. The protocol operation diverges, however, when a timeout event occurs and no ACK is received. To differentiate congestion from possible PU activity, the TFRC-CR at the source queries the FCC mandated spectrum database to check if a PU suddenly appeared on any of the feasible channels. Note that the source has no knowledge of the locations or the specific channels used by the nodes in the end to end connection. However, a sudden arrival event of the PU (as indicated by the database) and the timeout can be treated as correlated events, with a high probability. If this condition is true, the protocol enters into the *PU detected state*, and if not, the situation is interpreted as a normal case of network congestion. Additionally, the protocol continues to maintain an average ACK inter-arrival time (denoted as I_n), and the standard deviation of the round-trip time (denoted as RTT_{stddev}). These values are used in the subsequent states to influence the rate control mechanism.

B. PU detected state

This is an intermediate state implemented as an additional measure to verify that the last ACK timeout in the *Normal state* was in fact due to PU activity. On entering this state, the source waits for a period I_n for any incoming ACK and continuously polls the spectrum database. If no subsequent ACKs are received, and the database reveals that the PU is still present, then the protocol transitions into the *Paused state*. On the other hand, if an ACK does arrive in that time period, then the protocol returns to the *Normal state* as this implies that the intermediate nodes are not affected by the PU activity. In such a case, the ACK timer expiry was due to random channel or congestion errors.

C. Paused state

In this state, the PU is determined to be present, and is assumed to be responsible for disrupting the continuous data stream as it has occupied the spectrum. The challenge now is to identify when the transmission rate can revert back to a higher value, and this is obtained by polling the connection bandwidth with an occasional packet. When a portion of the spectrum is occupied by a PU, the link layer algorithms on the node pair on the affected link may either pause the transmissions altogether, or immediately try and identify an alternate spectrum for that link. Note that the source has no idea of which of these options are actually selected, as no intermediate node feedback is allowed. Thus, by increasing the transmission rate too early, the source risks added interference to the PU before it vacates the spectrum. Also, by delaying the rate increase after a spectrum change, the source is unable to efficiently use the available bandwidth of the connection. By simply monitoring the spectrum database (which can continue to show the PU as present), the source remains unaware of a local spectrum change. We have undertaken a substantial set of simulations and empirically identify the optimal polling rate as $X_{bps} = \frac{s}{6*RTT_{avg}}$ (Section II), i.e., the source will poll the connection every 6 average RTT times whether or not the nodes have switched the channel. In comparison, TFRC reduces the rate after each ACK timer expiry in half, until it reaches a rate of $\frac{s}{64}$, which sends out a packet every 64 seconds. This leads to slow reaction to both the sudden reduction in bandwidth when the PU starts affecting the traffic chain, and to the higher available bandwidth once the PU is out of the vicinity.

If an ACK is received in the *Paused state*, the network restores the rate to the last rate recorded in the *Normal state* and enters the *Resumed state*. TFRC-CR perceives this packet as an indication that the intermediate nodes have moved to a vacant spectrum and allows the rate to adapt accordingly. If no feedback packets are received during this period, indicating that the nodes have not switched the spectrum, TFRC-CR will enter the *Slow start state* immediately after the PU leaves the spectrum. The PU exit time is known by the aforementioned spectrum database.

D. Resumed state

TFRC-CR enters this state when an ACK is received while being in the *Paused state*. The protocol interprets this ACK arrival as an indication that the intermediate nodes have switched to a vacant channel and allows for the rate to adapt accordingly by restoring it to the last known rate in the *Normal state*.

The protocol stays in this state until the PU exits, at which time it enters the *PU Exit state*. Notice that TFRC-CR is not returning to the *Normal state* yet because the ACK that was received in the *Paused state* could be due to an intermediate node falsely mis-detecting the PU presence.

If the the nodes never mis-detect the PU presence, then the protocol will never enter this state because it will remain in the *Paused state*. The protocol enters the *PU Exit state* when the current active PU exits the vicinity. This time is scheduled based on the query results from the integrated FCC database which is known at the sender.

The average ACK inter-arrival time I_{PU} is calculated during this time period for use in the next state.

E. PU Exit state

In the *PU Exit state*, the goal is to determine whether a slow-start is required or not. TFRC-CR slow-starts if the rate at the time of the PU exit is relatively low in comparison to the rate recorded during the last *Normal state*. In other words, the ACK received during the *Paused state* was a result of a sensing error and a slow-start to probe for new bandwidth is required. Otherwise, the protocol quietly returns to the *Normal state* because the intermediate nodes have found a vacant spectrum and resumed transmission. The decision whether to slow-start is made based on the results obtained in Algorithm 1.

Algorithm 1 : is slow start required

let I_n be the average inter-arrival of ACKs at the sender in the Normal state.
let I_{PU} be the the average inter-arrival of ACKS at the sender during the Paused state.
let t be the time now.
let t_{PU} time last ACK received during Paused state.

```
if I_PU > (2 * I_n) OR t − t_PU > (3 * I_n) then
    return true
else
    return false
end if
```

In summary, Algorithm 1 checks if one of the following is true, based on empirical observations:

- *Case I*: If the average inter-arrival of ACKs during the *Paused state* is larger than twice the average inter-arrival of ACKs during *Normal state*.
- *Case II*: If the time elapsed since the last ACK received during an ongoing PU activity is larger than 3 times the average inter-arrival of ACKs during *Normal state*.

F. Slow start state

TFRC-CR enters *Slow-start* if the rate during *Resumed state* was slow according to Algorithm 1 or if the previous state was the *Paused state*, i.e., no ACKs were received in *Paused state*. Slow-start is used to quickly probe the new vacant spectrum for the maximum available bandwidth. TFRC-CR slow-starts by resetting the weights and variables of TFRC. This is done by having the source flag the next packet as a slow-start request packet (SSREQ). When the destination receives this packet, it resets its own loss rate p calculations (see Section II) and sends back a slow-start acknowledgement packet (SSACK) immediately. During slow start, the nofeedback timer is set to $RTT_{avg} + 4 * RTT_{stddev}$. We use this as a more accurate result than TFRC's default static $\frac{2*packetsize}{300}$.
Once the SSACK packet is received at the source, TFRC-CR returns back to the *Normal state*, thus completing the cycle.

IV. Performance evaluation

We simulate TFRC-CR over a multihop chain in ns2 as depicted in Figure 5. In our simulation, we use the Cognitive Radio Ad-Hoc Network (CRAHN) framework built by [3] and place four nodes in a straight line. In this framework, the nodes do not send CTS to RTS requests if they sense any PU activity. This leads to packets being queued at the node immediately before the PU region and eventually dropped due to retries or timeouts. We set our sensing period to 0.1 seconds and transmission period to 3.0 seconds [10]. The bandwidth of the channel at each hop is set to $2\,\mathrm{Mbps}$. The nodes in this simulation do not switch to another spectrum when PU is detected; they wait until the PU exits the spectrum to start transmitting again. All sensor nodes pick from 10 available spectrum bands at random at the beginning of the simulation. Each node will have two different interfaces: one for receiving packets and one for sending. We vary the type of PU activity from an exponentially distributed on and off times with mean (2 sec) and (10 sec) respectively, which we name *short bursts*. Similarly, a PU on time of 100 secs is designed to simulate *long PU activity* events.

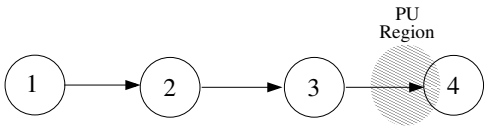


Fig. 5: 3-Hop chain and PU region

The simulations compare TFRC with TFRC-CR regarding the following metrics: (i) throughput over time, (ii) interference percentage with the PU, (iii) total data received by the receiver over the same period of time, (iv) the goodput of the data stream, and (v) the queue length of the affected node.

A. Throughput over time

Figure 3 and 4 compare TFRC with TFRC-CR in terms of throughput at the receiver for long and short activity durations of the PU, respectively. The simulation was run for for 300

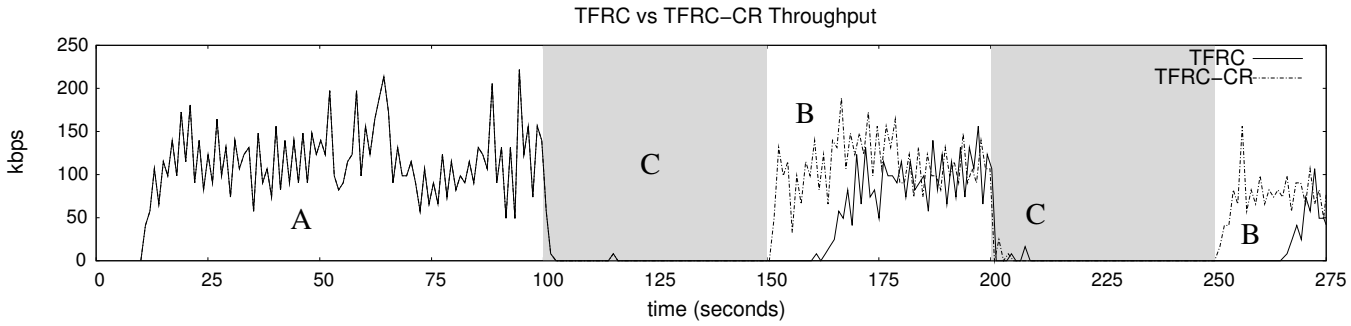


Fig. 3: Throughput (kbps) vs Time for 3-hop (long PU activity)

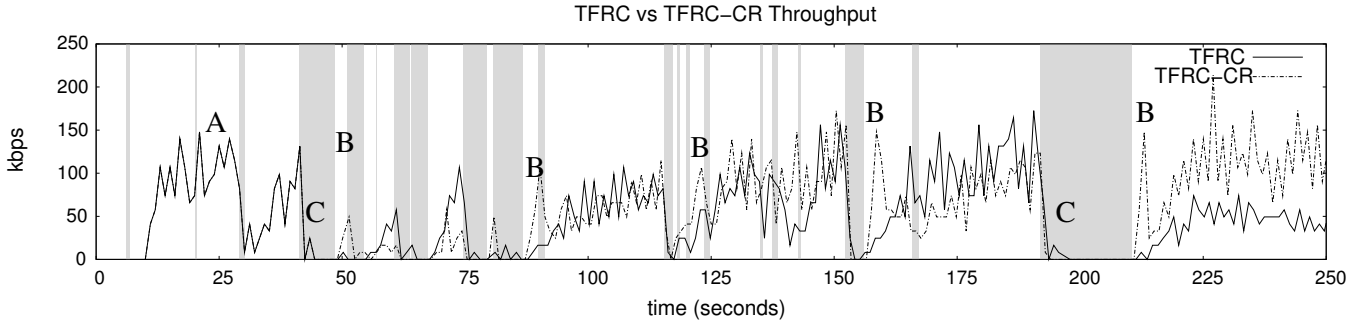


Fig. 4: Throughput (kbps) vs Time for 3-hop (short bursts)

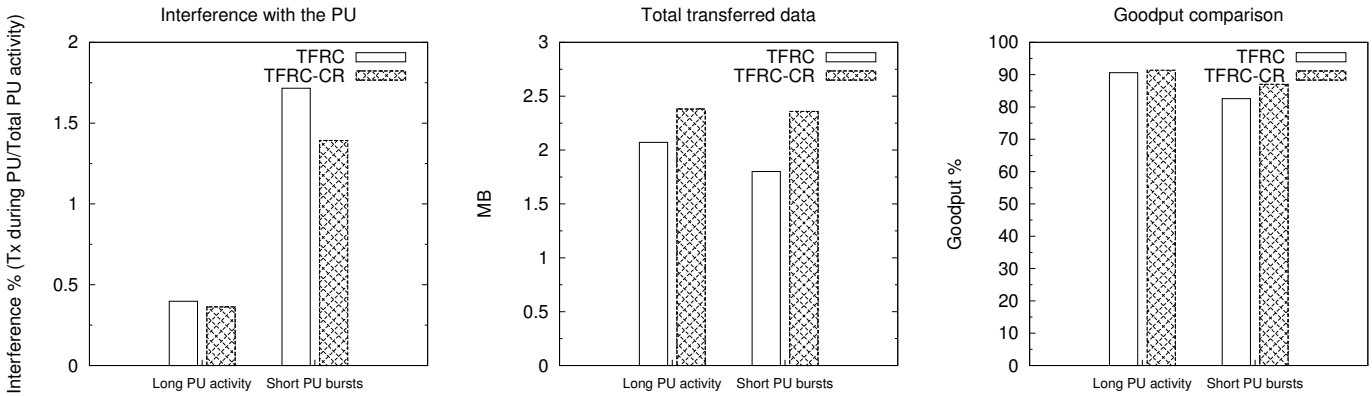


Fig. 6: The interference, total data transmitted and goodput comparison are provided in (a), (b) and (c), respectively.

seconds, and the regions of interest are denoted with letters A, B and C. The areas in gray represent the PU active regions.

1) Region A: In this region, TFRC-CR is in the Normal state. We can observe that the protocol's throughput matches that of TFRC. Interesting to note that in Figure 4, although the PU was affecting the traffic stream, TFRC-CR could not detect it because it did not encounter a nofeedback timer expiry during these short PU activity periods.

2) Region B: Areas indicated by B indicate that TFRC-CR goes into slow-start immediately after the PU exits the vicinity. We notice here that TFRC is unaware of PU activity, and due to the reduction in rate during the PU activity periods, the data flow resumes later. The longer the PU activity, the slower the rate, and hence, the later the protocol resumes. This is clearly indicated by Figure 3.

3) Region C: In this area, the PU is active but we notice the spikes in throughput during this period. The spikes occur more frequently at the beginning of the PU activity region due to the long time it takes TFRC to reduce the rate (i.e. with every ACK timer expiry, it reduces the rate by half). This slow reaction to the link disconnection caused by the PU causes undesirable interference. In these areas, TFRC-CR reduces the rate to $\frac{s}{6*RTT_{avg}}$ until an ACK is received. This leads to less interference as will be shown in the next section.

B. Interference with the PU

The immediate reduction in rate when TFRC-CR encounters the first ACK timer expiry leads to less interference with the PU. For example, with I_n of 1.2 sec, a rate of 3000 bps in the Normal state, and a nofeedback timer set at 2 seconds, TFRC-CR will reduce the rate to 416.67 bps in 2 sec. In comparison, the same rate will be reached by TFRC after 5.8 seconds leading to an excess of retransmission attempts during that time period. We calculate the interference to be the total time it takes to transmit RTS, CTS, ACK and DATA packets divided by the total PU on period. 6a shows the lower interference of TFRC-CR in both the short bursts and the long PU activity simulation runs.

C. Total data received

Due to the fact that TFRC fails to resume the data stream immediately after a PU exit or to use slow-start to probe for the new network bandwidth, TFRC-CR is able to transmit more data in the 300 seconds of the simulation with both types of PU activity. This can be seen in Figure 6b. In the short bursts scenario, the amount of increase is 33%.

D. Goodput

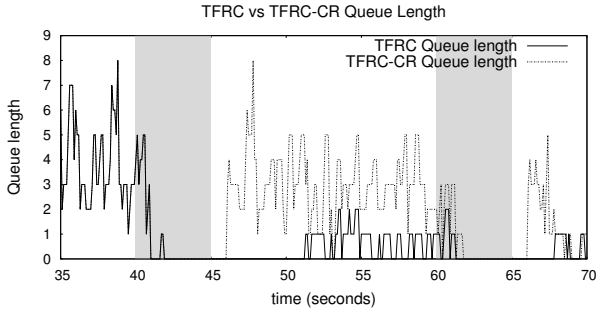
We calculate the goodput as $\frac{total\ data\ packets\ received}{total\ data\ packets\ sent}$. We can infer from Figure 6c that TFRC and TFRC-CR have very similar goodputs, however, TFRC-CR has slightly higher percentages due to it being able to slow down faster during the Paused state (read: when PU is active), and by doing that, it avoids the excess wasted packets that occur in TFRC.

E. Queue length

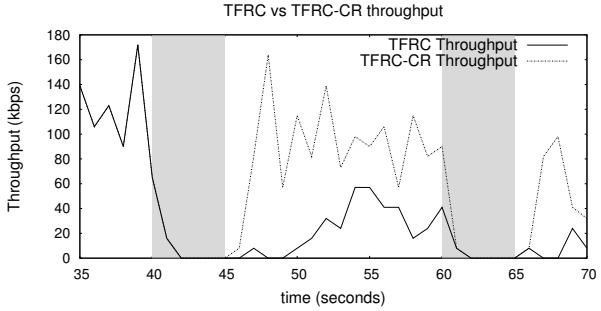
When the affected nodes detect PU activity, they halt any transmission that is scheduled at the MAC layer. This causes the queue to build up on the node immediately before the PU active region or node 3 in our topology. Figure 7a shows the queue lengths at that node. We note that there in no excessive build up on the queue and the increase in queue lengths are proportional to the sending rate at that time as shown in Figure 7b. When the PU enters the vicinity at time $60sec$, TFRC-CR's queue length are at most 1 packet higher than TFRC's even though the throughput is double that of TFRC.

V. Conclusion

We presented an equation-driven TCP protocol that is geared to meet the demands of CR networks, which also accommodates some of the latest developments in this space, such as spectrum database access. Our solution TFRC-CR is demonstrated to perform significantly better than its classical counter TFRC, with respect to both PU protection and transmission efficiency in a dynamically changing spectrum environment. Different from the few existing works at the transport layer for CR, our protocol does not assume any cross-layer feedback or other inputs from intermediate nodes. Our future work is aimed at changing the basic rate control equation based on different spectrum-related events.



(a)



(b)

Fig. 7: (a) Queue length of TFRC and TFRC-CR at node 3 and (b) the corresponding throughput of the network

VI. Acknowledgment

The leading author would like to thank Qatar University for the PhD scholarship support.

References

[1] K. R. Chowdhury, M. Di Felice and I. F. Akyildiz. TP-CRAHN: a transport protocol for cognitive radio ad-hoc networks. In *INFOCOM 2009, IEEE*, pages 2482-2490, April 2009.

[2] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," In *Proc. of ACM SIGCOMM, Aug. 2000*.

[3] Marco Di Felice, Kaushik Roy Chowdhury, and Luciano Bononi, "Modeling and performance evaluation of transmission control protocol over cognitive radio ad hoc networks," in *Proc. of ACM MSWiM*, 2009.

[4] Karthikeyan Sundaresan, Vaidyanathan Anantharaman, Hung-Yun Hsieh, and Raghupathy Sivakumar, "ATP: A reliable transport protocol for ad hoc networks," in *IEEE Trans. Mob. Comput.*, 4(6):588603, 2005.

[5] FCC, Second Memorandum Opinion and Order, ET Docket No. 10-174, September 2010.

[6] T. Goff et al., "Freeze-TCP: A True End-to-End TCP Enhancement," in *Proc. IEEE INFOCOM, 2000.*

[7] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. "WTCP: A reliable transport protocol for wireless wide-area networks," in *Wireless Networks,* 8(2-3):301-316, 2002

[8] G. Holland and N. H. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks." in *Proc. of ACM MOBICOM, pp. 219-230, Seattle, WA, Aug. 1999.*

[9] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, TCP selective acknowledgment options, RFC 2018 (October 1996).

[10] W. Y. Lee and I. F. Akyildiz, Optimal Spectrum Sensing Framework for Cognitive Radio Networks, *IEEE Trans. on Wireless Comm.*, vol. 7, no. 10, Oct. 2008.