Ad Hoc Networks xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

TFRC-CR: An equation-based transport protocol for cognitive radio networks

Abdulla K. Al-Ali *, Kaushik Chowdhury

Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115, USA

ARTICLE INFO

Article history:
Received 20 November 2012
Received in revised form 2 April 2013
Accepted 4 April 2013
Available online xxxx

Keywords:
Transport protocol
Cognitive radio
Congestion control
Equation-based
Sensor networks
Rate control

ABSTRACT

Reliable and high throughput data delivery in cognitive radio networks remains an open challenge owing to the inability of the source to quickly identify and react to changes in spectrum availability. The window-based rate adaptation in TCP relies on acknowledgments (ACKs) to self trigger the sending rate, which are often delayed or lost owing to intermittent primary user (PU) activity, resulting in an incorrect inference of congestion by the source node. This paper proposes the first equation-based transport protocol based on TCP Friendly Rate Control for Cognitive Radio, called as TFRC-CR, which allows immediate changes in the transmission rate based on the spectrum-related changes in the network environment. TFRC-CR has the following unique features: (i) it leverages the recent FCC mandated spectrum databases with minimum querying overhead, (ii) it enables fine adjustment of the transmission rate by identifying the instances of true network congestion, as well as (iii) provides guidelines on when to re-start the source transmission after a pause due to PU activity. TFRC-CR is evaluated through an extensive set of module additions to the ns-2 simulator which is also released for further investigation by the research community.

1. Introduction

Cognitive radio (CR) networks enable opportunistic use of available licensed spectrum to reduce the pressure on the unlicensed ISM bands in the 2.4 GHz and 5 GHz range. While the main functional blocks of spectrum sensing, switching, and sharing have experienced rapid strides over the past decade [1], work on higher layers of the protocol stack, such as the transport layer that is essential for realizing large scale practical deployments, remains in a nascent stage.

To date, the work on CR transport protocols has been based on the TCP window behavior, where the acknowledgment packets (ACKs) sent by the receiver determine the state of congestion within the network [2,15,21]. This self-clocking mechanism of TCP is highly susceptible to the observed round trip time. With periodic interruptions caused by the primary user's (PU) appearance or large scale bandwidth fluctuations, this mechanism by itself is unable to distinguish true congestion from PU induced spectrum changes. These works that directly adapt TCP for CR networks rely on comprehensive information from the underlying layers, as well as the intermediate nodes of the data path route. While there are distinct merits in a cross-layer approach, such a design violates the traditional end-to-end paradigm associated with the transport layer.

In window-based transport protocols, the problem of reliance on ACK timing is exacerbated in CR networks because nodes pause their 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) rendering the self-clocking nature ineffective. The frequency and reliance on the ACKs for window based transmissions also lead to reverse path performance

* Corresponding author.
E-mail addresses: al-ali.a@husky.neu.edu (A.K. Al-Ali), krc@ece.neu.edu (K. Chowdhury).

1570-8705/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.adhoc.2013.04.007

Please cite this article in press as: A.K. Al-Ali, K. Chowdhury, TFRC-CR: An equation-based transport protocol for cognitive radio networks, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.04.007

impact on the forward DATA path. In TCP, this can amount to 10–20% of the data stream rate as demonstrated in [4]. This paper presents a fresh perspective on the design of CR-specific protocols using an equation-based approach, wherein the concept of the congestion window in classical TCP is eliminated, and instead, an equation is devised as a function of the effective packet loss rate. This equation is not dependent on the time variance of the returning ACKs, and hence, the source transmission rate is less impacted by temporary disruptions in the flow.

The authors of [4] also report that CSMA/CA at the link layer results in bursty end-to-end flows when coupled with TCP at the transport layer. We independently verify this in Fig. 1 for a three node network where no congestion is introduced. The observed increase in TCP throughput may not only cause a potential adverse impact to the CR network through congestion, but also to the PUs by interfering with their packet delivery performance. Instead, the equation based TCP Friendly Rate Control (TFRC), a representative of the broader class of equation based transport protocol [3], remains stable, and in the absence of any other external stimulus, avoids the bursty transmissions seen in TCP.

Our approach towards transport protocol design for CR follows a new direction of using an equation based control, hitherto unexplored in the current literature. For this, we use the TFRC as the departure point. We not only adapt the state-action behavior of TFRC, but also modify the actual rate control equation leading to our new design for CR that we name as TFRC-CR. The main features of this new protocol are as follows:

- It allows the TCP source to integrate with designated spectrum databases, as mandated by the FCC in a recent ruling [5]. This limited (and required) interaction with the database totally removes any need of feedback from the intermediate nodes or from the underlying layers. Thus, TFRC-CR reverts back to the classical end-to-end paradigm associated with the transport layer.
- It intelligently polls the spectrum database only when needed, by identifying a possible PU arrival event based on the observed trend in packet losses, i.e., it does not consume the back-end system resources used for interacting with the database. Current regulations from FCC specify database polling at least once every 60 s for Mode I devices (more on that in Section 5), and our aim is to increase the access frequency only when a critical need is detected.
- It enhances the speed of response by distinguishing between spectrum change and true congestion. Hence, the transmission rate is almost never penalized unless the need is justified. Likewise, the rate of increase in the transmitted segments when new spectrum becomes available is much higher than that possible in the classical window based TCP, owing to the immediate effect of the rate equation.
- It modifies the TFRC rate control equation by changing the definition of the loss-event interval. This change allows the protocol to utilize the bandwidth more efficiently by having a higher and more accurate sending rate and throughput.

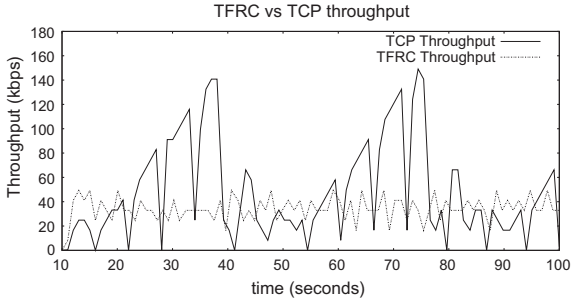


Fig. 1. Throughput comparison between TCP and TFRC in a 3-hop chain ad hoc network.

The rest of this paper is organized as follows: Section 2 gives the related works in the area of transport protocol design for CR. The preliminary background of TFRC and the motivation for adapting it for CRs is described in Sections 3 and 4. In Section 5, we describe the proposed protocol (TFRC-CR) in detail. Section 6 gives results from our comprehensive simulation study, and finally, we conclude our paper in Section 7 with pointers to future research.

2. Related work

While transport layer research in wireless networks has received considerable attention over the past decade, protocols focused specifically on CR networks are still in a nascent stage.

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 hand-off 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.

More specific to cognitive radio, various measurement studies have demonstrated the need for a new transport protocol for cognitive radio networks (CRNs) [13–15]. In particular, the suitability of TCP for CR networks, given its widespread use, has been explored in [13–17]. The work in [15] proposes modifications to TCP and introduces three different protocols: cogTCP, cogTCPE and cogTCPW. The *knowledge module* common to all of the above is linked to the transport protocol that leverages information from the link and physical layer such as sensing times and esti-

Please cite this article in press as: A.K. Al-Ali, K. Chowdhury, TFRC-CR: An equation-based transport protocol for cognitive radio networks, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.04.007

mated bandwidth. This family of protocols is designed for single hop scenarios.

Other protocols that leverage cross-layer information [18–20,22] also exist in literature. DSAsync [20] is a framework that modifies the base station's link layer that connects the outer wired network to the inner cognitive radio environment. It uses information from the link layer to explicitly pause the source and destinations' TCP streams, and is focused on a centralized (1-hop) topology between the CR nodes and the BS. Similarly to DSASync, [22] proposes modifications to the Base Station that connects TCP over the internet to a Cognitive Radio network by modifying the Base Station by two proposed methods: (a) Local loss recovery by base station and (b) Split TCP connection. The CR network is a one-hop network and the proposal restricts its modifications to the base station and not the transport protocol layer in the cognitive radio nodes. TP-CRAHN uses a window based approach similar to TCP, relies on intermediate node feedback and uses a cross-layer approach in each node involved [2]. TCP-CReno [21] modifies TCP Reno so that it pauses and resumes the data when the node is performing sensing. This information is retrieved from the MAC layer.

Protocols that change lower network layers to improve throughput at the transport layers were also investigated [18,19,23]. In [18], Luo et al. optimize the throughput of TCP by using an algorithm to decide which channel to use. Optimization at the physical, MAC, and link layer such as sensing times, access decision, modulation and coding scheme, and link layer frame size were also done in [19]. In [23], changes to the sensing and transmission times of the CR nodes were done to improve the throughput at TCP. These frameworks do not propose a new transport protocol, but improve the throughput by changing lower network layer parameters.

Different from all of the above, our aim is to design a transport protocol that is rate based, does not use information from underlying layers, operates through the end-to-end paradigm, and can work efficiently over multiple hops. These features allow the protocol to operate only at the source and destination. To the best of our knowledge, this is the first attempt at transport protocol design for cognitive radios with these specific goals. We are hopeful that will help in future practical deployment.

3. Discussion on rate control in TFRC

TFRC employs equation based 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 the additive increase multiplicative decrease behavior of TCP. Given that TFRC is rate based, we also have finer grain control over the sending rate. We begin by describing the classical rate control equation in TFRC.

Let x_i be the number of consecutive packets delivered to the destination in the i^{th} sample. The counting of these packets for the calculation of x_i continues till the first loss occurs, after the completion of the round trip time (RTT). Fig. 2 shows this procedure for two samples i and $i-1$.

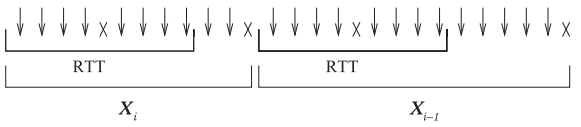


Fig. 2. Method of sample collection in TFRC: the first dropped packet after an RTT concludes a sample. A received packet is denoted by an arrow and a dropped packet by *x*.

Next, this step is repeated n times, obtaining $x_i, \ldots, x_n$ in this process. Now, the *loss event rate* (p) at the receiver is defined as the inverse of the weighted average of all x_i, $i = 1, \ldots, n$. TFRC, by default, averages the last 8 samples (i.e., $n = 8$) which gives an approximate snapshot of how the traffic flow changed over the last n RTTs. The weights w_i that are used to scale the x_i values are obtained as follows:

$$w_i = \begin{cases} 1, & i < \frac{n}{2}, \\ 1 - \frac{i-\left(\frac{n}{2}-1\right)}{\frac{n}{2}+1}, & \text{otherwise}. \end{cases} \quad (1)$$

As an example, if $n = 8$, then the values of the weights are: $\{1,1,1,1,0.8,0.6,0.4,0.2\}$. These weights are then used to calculate the weighted average loss interval (I_{mean}) of the last n samples $x_1, \ldots, x_n$ as:

$$I_{mean} = \frac{\sum_{i=0}^{n} x_i \cdot w_i}{\sum_{i=0}^{n} w_i}. \quad (2)$$

Finally, the loss event rate value (p) is obtained as $\frac{1}{I_{mean}}$ and informed to the sender through ACK packets. The sender changes the transmission rate through the Eq. (3) that estimates TCP's average sending rate (to achieve fairness with TCP):

$$X_{bps} = \frac{s}{RTT \cdot \sqrt{\frac{2bp}{3}} + \left\{ t_{RTO} \cdot \left(3\sqrt{\frac{3bp}{8}} \cdot p \cdot (1 + 32p^2) \right) \right\}}, \quad (3)$$

where X_{bps} is TCP's average transmit rate in bytes per second, s is the packet size in bytes, RTT is the round-trip time in seconds, $p \in [0,1]$, t_{RTO} is TCP's retransmission timeout value in seconds, and b is the maximum number of packets acknowledged by a single TCP ACK. By default, b is set to 1 and t_{RTO} is set to $4 \times RTT$. This choice is beyond the scope of our discussion and is discussed in detail in [3].

4. TFRC-CR design goals

In this section, we shall identify the specific features of classical TFRC that we target in our design of the transport layer protocol for CR networks.

4.1. Low utilization of available bandwidth

TFRC reduces the sending rate at the source whenever packets are dropped in the sending sequence. This simplistic approach is sufficient for wired networks, where packets are only dropped due to congestion, but not in CR networks where drops can happen due to multiple factors. We explain the mechanism as follows, and identify possible ways to adapt this situation.

Please cite this article in press as: A.K. Al-Ali, K. Chowdhury, TFRC-CR: An equation-based transport protocol for cognitive radio networks, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.04.007

Eq. (3) can be reduced to:

$$X_{bps} = \frac{s}{RTT \cdot f(s)}, \quad \text{where} \quad f(s) = \sqrt{\frac{2p}{3}} + 12 \cdot \sqrt{\frac{3p}{8}} \cdot p \cdot (1 + 32 + p^2). \tag{4}$$

When $t_{RTO} = 4 \times RTT$ and $b = 1$, these are approximated to equal TCP's sending rate [3].

From Eq. (4), the two contributing factors to the sending rate are *RTT* and *p*, the loss event rate. As explained earlier *p* is the inverse of the weighted average of the samples x_i (see Eq. (2)). In wireless networks, the sample x_i values tend to be smaller than in wired networks. This is because TFRC assumes that dropped packets occur due to congestion only (when intermittent losses are possible owing to channel errors). As soon as a single dropped packet is encountered at the receiver after an RTT has elapsed, the sample round *i* is completed, the value of x_i is logged and computation for x_{i+1} begins immediately. Fig. 2 describes this situation. The overall effect of this behavior is that for wired networks, this leads to relatively equal sample lengths, i.e., $x_1 \approx \cdots \approx x_n$ values. However, and particularly in CR networks, the disparity in the samples x_i is wider, as PU activity, spectrum availability changes, among other factors contribute to occasional packet losses. Fig. 3 shows this divergent behavior for wired and wireless cases. Because of this discrepancy in sample lengths in cognitive radio networks, TFRC's sending rate is fluctuating even when no stress events are introduced in the network. Our aim is to equalize the sample lengths by introducing a time-based sample collection method. This will produce equal lengths when the network is in normal condition and will also adapt in size as the network's available bandwidth changes.

4.2. Low transmission rate after PU departure

Classical TFRC is unable to use the maximum allocated bandwidth after a PU vacates the spectrum due to the transmission source's low rate that is caused by timeout events when a CR node ceases transmission or due to interference caused by PU activity. Instead, TFRC-CR is designed to neglect the last *n* loss event rates after a prolonged idle state due to PU activity. This prevents the pitfall of resuming the transmission with a false *p* which leads to less than optimal throughput. Note that classical TFRC will recover to the maximum transmission rate after at least *n* samples have been recorded. This takes at least *n* RTTs to achieve.

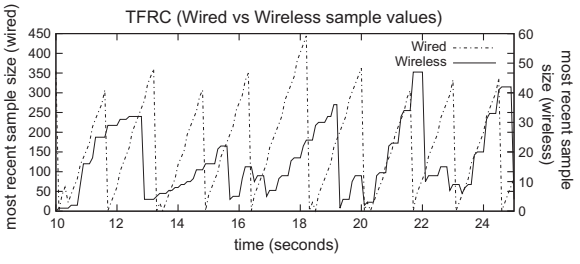


Fig. 3. Wired vs. wireless most recent sample (x_i) size over a two-hop network topology.

4.3. Slow recovery and ramp up

After an extended period of packet losses that limit TFRC's sending rate to the minimum rate, the protocol starts polling for changes in bandwidth over large intervals of time. This polling interval is a function of the current transmission rate. For example, during PU Activity, classical TFRC's *nofeedback* timer expires multiple times which leads to reduction in the effective rate by half each time until the minimum rate of $\frac{s}{t_{mbi}}$ is reached. Here, *s* is the packet size and t_{mbi} is set to 64 s in the default implementation. This means that TFRC will poll the network once every 64 s, which can cause an equivalent delay for the CR network to resume transmission again after a PU departure. Thus vital spectrum opportunity is wasted in this extended downtime for the CR network. Our goal is to speed the resurgence of the data flow by integrating the transport protocol with the FCC database which can inform the sending node in advance the time at which the current PU activity ends.

4.4. Buffer overload and interference

TFRC may send multiple packets during the duration of the PU activity as part of its regular rate control which causes additional interference with the PU. This serious problem multiplies at the link layer, where typically the medium access control protocol attempts several rounds of transmission per packet before reaching the maximum retry limit. Moreover, the added processing tasks as well as the consumption of buffer space by these additional packets that cannot be transmitted on the same channel contribute to the overhead. While the CR network can be designed to seek out and immediately leverage alternate channels, this feature is not known at the source in advance (owing to lack of inter-layer communication). By relying on the FCC database to inform the sender of possible disruptions due to PU activity, this problem can be alleviated; the sender will pause the data flow based on that information which will lessen the stress on the intermediate nodes' queues.

5. Design and Implementation of TFRC-CR

We first present the modified finite state machine for TFRC-CR which we will refer to as we describe the overall functions related to spectrum management and PU avoidance. Then, we discuss how to change the rate equation in TFRC-CR such that the connections utilize the maximum allocated bandwidth.

Note about FCC database and connectivity: In [5], the FCC has recently allowed different devices to use a centralized database to infer PU activity. Devices that are allowed to use the channels are categorized in three modes: Mode I, Mode II and sensing mode. Mode II are geolocation capable devices that are required to access the database directly at

Please cite this article in press as: A.K. Al-Ali, K. Chowdhury, TFRC-CR: An equation-based transport protocol for cognitive radio networks, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.04.007

least once per day before utilizing any channel, Mode I devices needs to connect to Mode II devices directly or a fixed-based station, and are required to refresh their local database information once every minutes. Sensing mode nodes need to be certified by the FCC and can then sense and use what they perceive as vacant spectrum. In the design of TFRC-CR, we are only interested in spectrum awareness at end locations and do not involve the intermediate nodes. This still maintains the end-to-end sense of the transport protocol. In any case, we do not rely on the knowledge of specifically "where" or "at which node" PU activity occurred in the connection, but that it occurred somewhere in the connection. Similar assumptions on rough estimations of a region of interest have been made in earlier works on routing (see Ref. [10] for example).

5.1. TFRC-CR spectrum management

This section covers the following key features of our proposed protocol: (i) ensuring that connection immediately becomes active after the PU leaves the impacted region, (ii) striking a balance between polling the network too frequently by the source, an action that may itself cause interference with the PU, and conversely, reacting too slowly to spectrum change, and (iii) estimating the available bandwidth as soon as possible after the PU vacates the spectrum. These changes are explained below using a finite state machine diagram, shown in Fig. 4.

5.1.1. Normal state

This is the default state of TFRC-CR. The protocol returns to this state whenever the source infers the connection to be free of spectrum outages. We describe in detail how the equation governing the sending rate (and hence, congestion control) in this state is modified in Section 5.2. Recall that when a packet is dropped, the *no feedback* timer expires in the absence of the ACK. From here onward, our protocol's operation diverges from classical TFRC: To differentiate congestion from possible PU activity at this timer expiry, the source queries the FCC mandated spectrum database to check if a PU arrived on any of the feasible channels. Note that the source has no knowledge of the location of the nodes in the connection except the destination, nor the specific channels used by any of these nodes. However, a sudden arrival event of the PU (as indicated by the database) and the resulting timeout are treated as correlated events. If the database affirms the PU presence, the protocol enters into the *PU detected state*, and if not, the ACK loss is interpreted as a case of network congestion and is handled by the standard TFRC rate control which cuts the sending rate in half. In the *normal state*, the protocol continues to calculate the 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.

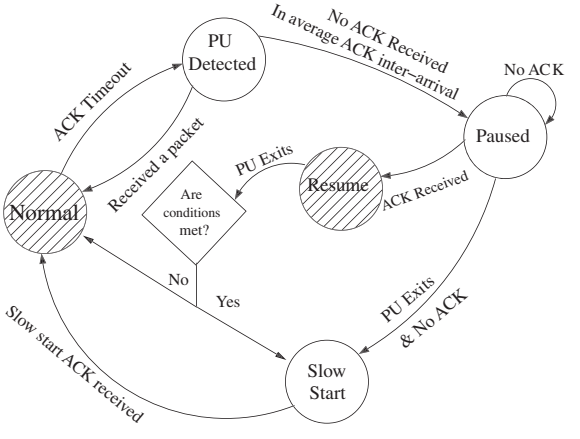


Fig. 4. TFRC-CR finite state machine. Shaded states involve rate modifications discussed in Section 5.2.

Please note that our reliance on the ACK timer expiry to query the FCC database and infer PU activity here is different than congestion inference that is typically used in window-based transport protocols; TFRC-CR's end-destination periodically generates ACKs for the sender, whether or not (i) there is true network congestion, and (ii) packet arrival to the destination. Thus, the rate control is decoupled from the frequency of returning ACKs, which is the typical method used in window-based protocols. In TFRC-CR, if a sender does not receive an ACK, then it signals a larger event, such as a total disruption on the link. In this state, we use this scenario to map a PU appearing which renders the link unusable.

5.1.2. PU detected state

This intermediate state is implemented as an additional measure to verify that the last ACK timeout in the *normal state* was in fact due to PU activity. Upon entering this state, the source waits for a period (the average inter-arrival time during normal state or I_n) for any incoming ACK, while continuously polling 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* where the source must wait for the PU activity to get completed. 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 errors or congestion.

5.1.3. Paused state

In this state, TFRC-CR determined that the PU is present and assumed that it is responsible for disrupting the continuous data stream. 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 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 which of these options is selected as no intermediate node feedback is allowed. Additionally, simple monitoring of the spectrum database, even if it indicates the presence of a

Please cite this article in press as: A.K. Al-Ali, K. Chowdhury, TFRC-CR: An equation-based transport protocol for cognitive radio networks, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.04.007

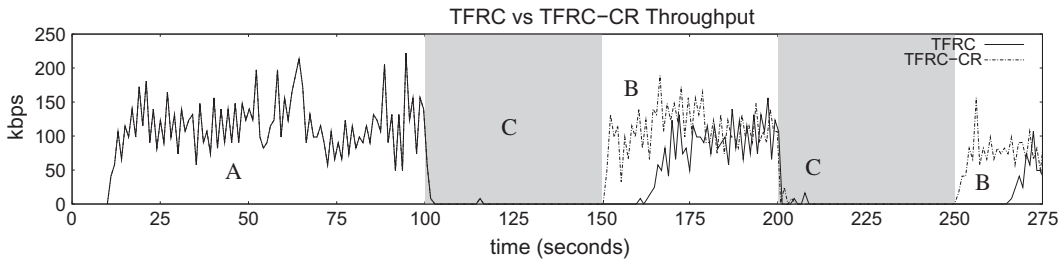


Fig. 5. Throughput (kbps) vs. Time for 3-hop (long PU activity).

PU, does not reveal any corrective action by the nodes in the connection. Thus, determining when the connection is active again is a non-trivial task.

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, the source is unable to efficiently use the available bandwidth of the connection if the PU vacates the spectrum earlier or the nodes in the connection transition to a new channel. We have undertaken a substantial set of simulations and empirically identify the optimal polling rate as $X_{bps} = \frac{s}{RTT_{avg}+(4\times RTT_{std})}$ (Section 3), i.e., the source will send a packet every time the Retransmission Timeout (RTO) value of TCP [12] expires 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 $\frac{s}{64}$ (see Section 4.3), which sends out a packet every 64 s. This leads to slow reaction to both the sudden reduction in bandwidth when the PU starts affecting the connection, and to the higher available bandwidth once the PU is out of the vicinity.

If an ACK is received in the *paused state* due to the polling packets that the sender transmitted, the protocol enters into the *resumed state*. TFRC-CR perceives this ACK 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 querying the aforementioned spectrum database.

5.1.4. Resumed state

During PU activity and while the protocol is in the *paused state*, the intermediate nodes may either switch to a vacant spectrum or remain in the occupied channels. If the intermediate nodes have switched spectrum and the link is no longer disrupted, the sender will receive an ACK from the destination. TFRC-CR will then enter the *resumed state* and initiate slow-start (see *Slow start state* below for details). This is done to adjust the sending rate based on the new channel characteristics that the nodes have switched to. Please note that the rate control at this state is modified according to the discussion that will follow in Section 5.2.

The protocol stays in this state until the PU exits, at which time it runs an algorithm to see whether another slow-start is required. Notice that TFRC-CR does not yet return to the *normal state* because the ACK that was received in the *paused state* could be due to an intermediate node falsely misdetecting the PU presence; a realistic outcome considering the existing state-of-the-art sensing algorithms.

If the nodes never misdetect the PU presence and have not switched to a vacant spectrum, then the protocol will not enter this state because it will remain in the *paused state*. The protocol runs Algorithm 1 when the current active PU exits the vicinity. This time is scheduled based on the query results from the integrated FCC spectrum database which is known at the sender. Finally, the average ACK inter-arrival time I_{PU} is calculated in the duration of this state for use in Algorithm 1.

5.1.5. Slow start decision box

The goal in this algorithm 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 immediately 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 average inter-arrival of ACKs at the sender during the *paused state*
let current time = t
let t_{PU} time last ACK received during *paused state*
1: **if** $I_{PU} > (2 \times I_n)$ OR $t - t_{PU} > (3 \times I_n)$ **then**
2: return *true*
3: **else**
4: return *false*
5: **end if**

In summary, Algorithm 1 checks if either of the following is *true* based on empirical observations to correctly determine whether the packet received during slow-start was in error and TFRC-CR should therefore slow-start:

Please cite this article in press as: A.K. Al-Ali, K. Chowdhury, TFRC-CR: An equation-based transport protocol for cognitive radio networks, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.04.007

- *Case I*: If the average inter-arrival of ACKs during the *paused state* (I_{PU}) is larger than twice the average inter-arrival of ACKs during *normal state* (I_n).
- *Case II*: If the time elapsed ($t - t_{PU}$) since the last ACK received during an ongoing PU activity is larger than 3 times the average inter-arrival time (I_n) of ACKs during *normal state*. This is necessary because the average ACK inter-arrival time (I_n) is calculated online, and if there are two consecutive ACKs that arrive due to a PU misdetection, I_n will be small. The second condition is designed to catch these exceptions.

5.1.6. 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 the *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 3) and sends back a slow-start acknowledgement packet (SSACK) immediately. During slow start, the *nofeedback timer* is set to $RTT_{avg} + 4 \times RTT_{stddev}$ [12] where RTT_{avg} and RTT_{stddev} are the average and standard deviation of the round-trip-time. We use this as a more accurate result than TFRC's default static $\frac{2 \times packetsize}{300}$.

Once the SSACK packet is received at the source, TFRC-CR returns back to the *normal state*, thus completing the cycle.

5.2. Sending rate adaptation

As explained in Section 3, TFRC ends the collection of each sample whenever it encounters a dropped packet. However, in wireless networks susceptible to random errors [4], this leads to sub-optimal and small sample (x_i) values that are far from the correct ones that allow for full utilization of the available bandwidth in the wireless network.

Due to the random nature of these dropped packets, our protocol cannot rely on them to determine the correct size of the samples. Instead, we propose to look at a time-based window for incoming packets (see Fig. 6). This way, our protocol ends sample collection after a given period of time instead of when it encounters the first dropped packet. This method is further explained by identifying how to select the interval for collecting the samples, and how to scale the observed samples in that interval as a function of the connection length.

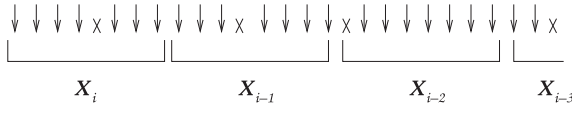


Fig. 6. Method of sample collection in TFRC-CR; instead of relying on the first dropped packet, sample collection is based on a static time interval.

5.2.1. Collection time interval

The collection time interval needs to strike a balance between being prohibitively high, and therefore reacting slowly to network condition changes (e.g. bandwidth increase/decrease, PU activity, congestion), and too low such that the collected samples do not represent the correct network condition in terms of the amount of packets that arrived and what percentage of those were dropped. In our empirically obtained results for centralized networks (where the source and destination are directly connected), setting the collection time interval to 1.0 s displayed a sound balance between having a good reaction speed to condition changes and having enough packets in the samples (x_i) to meet the transmission rate that would fully utilize the spectrum's available bandwidth.

Furthermore, we observed for larger connections, i.e., for 3-hops and more, the packets received in the 1.0 s duration were too few to correctly represent the correct sending rate at the sender. Simply increasing the collection time interval is not possible due to the adverse effect on the speed of the response to traffic congestion or spectrum related changes. Thus, we add an initialization phase that precedes the slow-start phase of the connection, wherein the source–destination pair send test packets to identify a static multiplier M. This multiplier is a function of the length of the connection that is unknown to the source and must be empirically decided. In the actual operation, the source scales the number of correctly transmitted samples in the 1.0 s duration with this value M before weighting their average and determining the effective loss rate p.

5.2.2. Initialization phase for choice of multiplier

The choice of a multiplier is critical to have the best balance between having a very high sending rate, which can adversely lead to higher RTT – This is because the increase in queuing delay at the intermediate nodes may eventually result in higher dropped packet rates when the sending rate is significantly higher than the capacity of the connection – and conversely, having a very small multiplier value leads to low throughput and under-utilization of the available bandwidth. Eq. (2) therefore becomes:

$$I_{mean} = \frac{\sum_{i=0}^{n} x_i \times w_i \times M}{\sum_{i=0}^{n} wi}, \tag{5}$$

where M is the multiplier value.

Fig. 7 shows the effect of increasing the multiplier value (The exact network parameters are discussed in details in Section 6). As M increases, the RTT increases and eventually, the dropped packet rate increases when the buffers overflow in the intermediate nodes. For the 3-hop scenario, ≈190 is the best multiplier value, and any higher leads to unnecessary increase in RTT (queuing delay), higher dropped packet rate without any significant throughput gain. Likewise, for a 4-hop topology, the correct value is ≈390. Note the high dropped rate when the multiplier is very low which is due to the very low throughput.

When TFRC-CR slow starts, we find the optimum M by using Algorithm 2. This algorithm increments M until the change in throughput increase is less than %10 (condition $\frac{TP_{now}}{TP_{prev}} > 1.1$) while making sure that the dropped rate re-

Please cite this article in press as: A.K. Al-Ali, K. Chowdhury, TFRC-CR: An equation-based transport protocol for cognitive radio networks, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.04.007

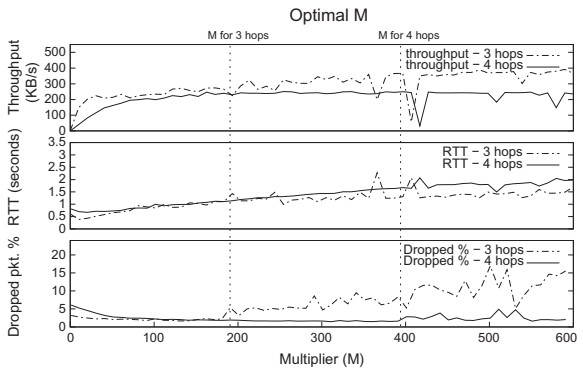


Fig. 7. Optimal multiplier M value for a 3 and 4-hop topologies.

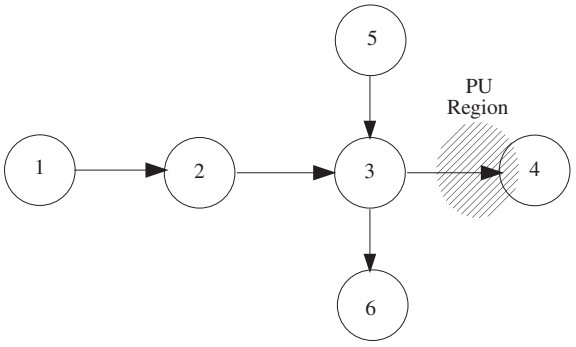


Fig. 8. 3-Hop chain and PU region.

mains below %10 (condition *dropped_rate* < 0.10). This will give us the highest possible throughput while maintaining a low dropped rate. The *while loop* (lines 5–8) are traversed once every time a new ACK is received at the source.

Algorithm 2. Finding optimum *M*

1: Let TP_{now} be throughput at current time
2: Let TP_{prev} be the previous throughput
3: Let *dropped_rate* be the dropped packets' rate
4: $M = 1$ and $TP_{prev} = 1$
5: **while** *dropped_rate* < 0.10 and $\frac{TP_{now}}{TP_{prev}} > 1.1$ **do**
6: M + =1
7: $TP_{prev} = TP_{now}$
8: **end while**

This optimal value of *M* is retained for all future scaling during the protocol operation.

6. Performance evaluation

In our simulation, we use the Cognitive Radio Ad-Hoc Network (CRAHN) framework from [13], and expand it significantly to support the transport layer operations. The revised simulator which incorporates TFRC-CR can be downloaded from the link in [24] where instructions to compile and integrate it with existing ns-2 installations can be found. In Sections 5.1 and 5.2, we simulate TFRC-CR over a multihop chain in ns-2 as depicted in Fig. 8. This topology is best suited to evaluate our protocol under the following different scenarios: (a) single and multihop simulations where node 4 is the sink and the sending node can vary from node 1 to node 3, (b) create a bottle neck at node 3 either due to spectrum bandwidth change, or (c) due to congestion (when node 5 has a second active connection to node 6).

In the simulation setup, nodes ignore incoming RTS packets if they sense any PU activity within their immediate vicinity. This leads to an increase in queued packets at the node immediately preceding the nodes in the active PU region and eventually dropped due to retries or timeouts. Each node has a 0.1 probability of misdetecting the PU activity. In these cases, the nodes that misdetect, transmit data concurrently with the PU causing interference. We set our sensing period to 0.1 s and transmission period to 3 s [11]. The transmission rate at the link layer is set to 11 Mbps using 802.11b specifications. The nodes in this simulation do not switch to another spectrum when the PU is detected; they wait until the PU exits the spectrum to start transmitting again. This is done to clearly demonstrate how the source correctly detects, pauses and resumes the transmission. All 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. The authors of classical TFRC in [3] recommend setting *b*, the number of packets that are acknowledged by a single ACK, to 1 and *n*, the number of weights to average, to 8. The authors advise against setting *n* to a higher number owing to the slow reaction to congestion conditions that this would ensue. We follow these recommendations throughout this section.

We will first illustrate our spectrum management changes (Section 5.1) followed by our rate adjustment evaluation (Section 5.2). Lastly, we evaluate our protocol in high density node and PU scenarios in Section 6.3. Our simulation compares TFRC-CR against TCP and default TFRC as the baseline window and equation based transport protocols, respectively. To the best of our knowledge, there are very limited existing transport protocols specifically targeting cognitive radios (for e.g., TP-CRAHN [2], cogTCP [15] and TCP-CReno [21]). Unfortunately, comparisons with them are not viable due to (i) completely different window-based and rate-based implementations, (ii) extensive feedback from the underlying layers and intermediate nodes that is required in them, but specifically avoided in our approach, and (iii) the fact that some of proposed protocols (e.g. cogTCP [15]) are designed for a centralized wireless transmission topologies while our protocol is designed to work in multihop scenarios as well.

6.1. Spectrum management

In this section, we omit our changes to the rate control in order to showcase the state machine control algorithm (Section 5.1). Fig. 5 shows us the throughput difference between these two protocols in one simulation run. Areas in

Please cite this article in press as: A.K. Al-Ali, K. Chowdhury, TFRC-CR: An equation-based transport protocol for cognitive radio networks, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.04.007

gray denote PU activity regions. Throughput regions *A*, *B* and *C* are of interest. This simulation is run over a 3-hop chain (i.e. Node 1 sends to node 4 while nodes 5 and 6 are deactivated in Fig. 8).

- *Region A*: In this region, TFRC-CR is in the Normal state. We observe that the protocol's throughput matches that of TFRC.
- *Region B*: We see that immediately following the PU activity region that ends at time 150 s and 200 s, TFRC-CR slow-starts immediately because it utilizes the information received from the spectrum database. Due to the rate reduction that happens in TFRC during the PU activity time, the data stream resumes later. The longer the PU activity, the slower the rate which leads to longer delays.
- *Region C*: In this region, the PU is active but we notice the spikes in throughput. 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 timeout, it reduces the rate by half). This problem is exacerbated when the nodes surrounding the PU region (nodes 3 and 4 in Fig. 8) misdetect the PU activity, which occurs with a probability of 0.1. To clearly demonstrate the effects of our *paused state* adjustment, however, one must look at the sending rate during this time, and not the throughput at the receiver. This is discussed in the next section.

6.1.1. Sending rate during paused state

Fig. 9 plots the sending rate in the first PU region from Fig. 5. We observe that a) TFRC-CR reduces the rate to the *paused state* polling rate (i.e., $\frac{s}{RTT_{avg}+(4\times RTT_{std})}$) when the PU is affecting the region, and b) how the sending rate slow-starts immediately after. This is in contrast with the slow decay that happens in TFRC, followed by the slow resumption of the rate after the PU exits.

6.2. TFRC sending rate adjustment

In this section, we illustrate the advantages of our rate adjustment (Section 5.2) by comparing the throughput, interference with PU and queue lengths of the affected nodes.

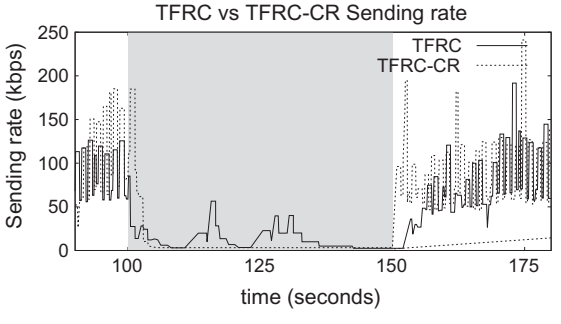


Fig. 9. Sending rate X_{bps} during PU activity region.

6.2.1. Throughput with variable PU on time and number of hops

First, we look at how the throughput changes as we modify the PU on time in centralized and in multihop (2 and 3 hop) scenarios. For each data point, we run the simulator using 20 different seeds and compute the average throughput. The PU on and off time follows an exponential distribution and the average value are denoted by the *x*-axis. The topology follows that of Fig. 8, where node 4 is always the receiver and the role of the sender alternates from nodes 1 to 3, depending on the hop count.

Centralized system: Fig. 10 compares TCP, TFRC and TFRC-CR in a centralized (1-hop) network. Interestingly, TCP and TFRC perform well in such topologies due to less random channel errors and dropped packets. We can see that TFRC-CR's performance is relatively close to that of TCP and TFRC. When the PU on activity is small, TFRC-CR performs worse due to the spectrum management control where TFRC-CR pause the rate every time it detects PU activity and resumes afterwards. Meanwhile, TFRC and TCP do not pause during these periods, and because the PU activity is small, the nodes that are immediately affected by the PU queue the incoming packets at the MAC layer and send them out once PU activity seizes. We do notice also that TFRC-CR outperforms TCP and TFRC as the PU on time increases.

Multiple (2 and 3) hops: Fig. 11 gives us the average throughput for both TCP and TFRC in 2 and 3 hop scenarios. We notice the low throughput for both protocols, which further degrades to 0 as PU on time increases. TFRC-CR's 2-hop and 3-hop implementations are plotted in Fig. 10, due to the large difference in scale. We can clearly see the advantages of using a time-based window to compute the loss event rate *p*. In the 2-hop scenario with an average PU on time of 1 s, we see that TFRC-CR's average throughput is 100 KBps while TFRC averages at 8 KBps and TCP at 12 KBps. This is about a 10 times improvement over both traditional protocols. This example clearly demonstrates the ineffectiveness of existing transport protocols for multihop end-to-end CR scenarios.

6.2.2. Throughput performance under different stress events

Fig. 13a and b compare the three transport protocols under different stress events. The three stress events are (a) congestion: node 5 sends to 6 in our topology (Fig. 8)

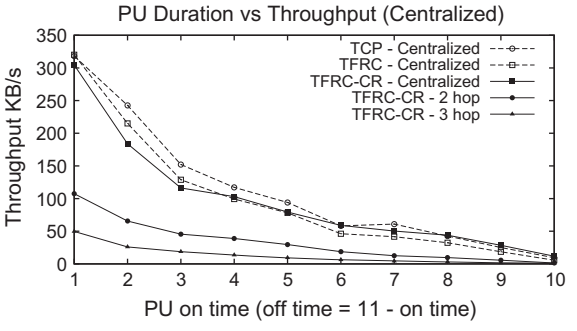


Fig. 10. Throughput comparison as PU on time increases for different number of hops.

Please cite this article in press as: A.K. Al-Ali, K. Chowdhury, TFRC-CR: An equation-based transport protocol for cognitive radio networks, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.04.007

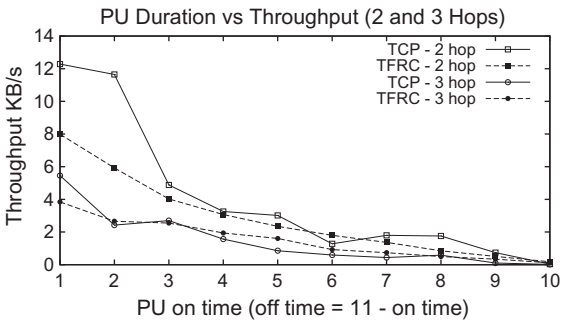


Fig. 11. Throughput comparison as PU on time increases for 2 and 3 hop scenarios (TCP & TFRC).

using the same transport protocol, (b) bandwidth increase: we set the hop from node 3 to 4 to be a bottle neck by setting its bandwidth to 5 Mbps, and (c) bandwidth decrease: we set the bandwidth of the aforementioned hop to 1 Mbps.

This simulation is run for a 2-hop scenario (i.e. node 2 sends to node 4 in our topology). Twenty different simulations are run and averaged for each bar. The three stress scenarios are introduced at second 100 in the simulation run and the network returns to normal at second 200. The results are averaged in that stress interval (second 100 to 200).

Though TFRC-CR's throughput outperforms TFRC and TCP in all scenarios, however this comes at a cost: the queue length at node 3 (the node immediately before the sink node) is higher for TFRC-CR than TFRC or TCP (Fig. 13b) with the exception of the bandwidth increase scenario where it compares similarly to TCP. The increase in queue lengths is expected as this is a direct result of having a larger throughput at the bottleneck node leading to accumulated packets at the queue when these stress events are introduced.

6.2.3. Interference with the PU

This performance analysis compares the three transport protocols in regards to interference with the PU. We calculate the interference percentage with the PU activity by measuring the sum of time it takes to transmit RTS, CTS, ACK and DATA packets during PU activity divided by the total PU activity time period. For each transport protocol, a 2-hop scenario is simulated 20 times and averaged with the variable PU on rate indicated by the *x*-axis.

TFRC-CR's interference is notably higher than TFRC and TCP (Fig. 12a). This is attributed to the much higher throughput for TFRC-CR (Fig. 12b) in the same simulation runs: when the sending rate is high, the node immediately before the PU vicinity (node 3) queues packets that are on route and then sends them out when it misdetects PU activity which we set at a probability of 0.1. This happens regardless of how the sender pauses the sending rate. A solution to this problem is to lessen the probability of misdetection by having more accurate sensors or to immediately empty the queue if a PU is encountered. This involves changing the MAC layer protocol which is beyond the scope of this paper.

Note that although the interference bars look high for TFRC-CR, the percentage is well below 3% even for the worse case (low PU on time).

6.2.4. Goodput

Goodput is measured as $\frac{totaldatapacketsreceived}{totaldatapacketssent} * 100$. We can see from Fig. 12c that for the same simulation runs as those of Fig. 12a and b, the goodput of TFRC-CR is 95% when PU activity is small (PU time is 2 and 20 for on and off times) and better than TFRC and TCP when PU activity is high (10 and 5 s for PU on and off time, respectively). This is an improvement especially considering that TFRC and TCP had little to no throughput in the high PU activity scenario. TFRC-CR's higher goodput rates at high PU activity scenarios is a direct result of having TFRC-CR slow the sending rate in the *paused state* (i.e., when PU is active). By slowing down immediately, it avoids the excess, and eventually lost, packets that are sent in TCP and TFRC.

6.3. High density nodes and PUs

In the following simulation, we place 25 nodes in a 5×5 grid and we vary the number of end-to-end streams and PUs. Our purpose is to simulate a high density node

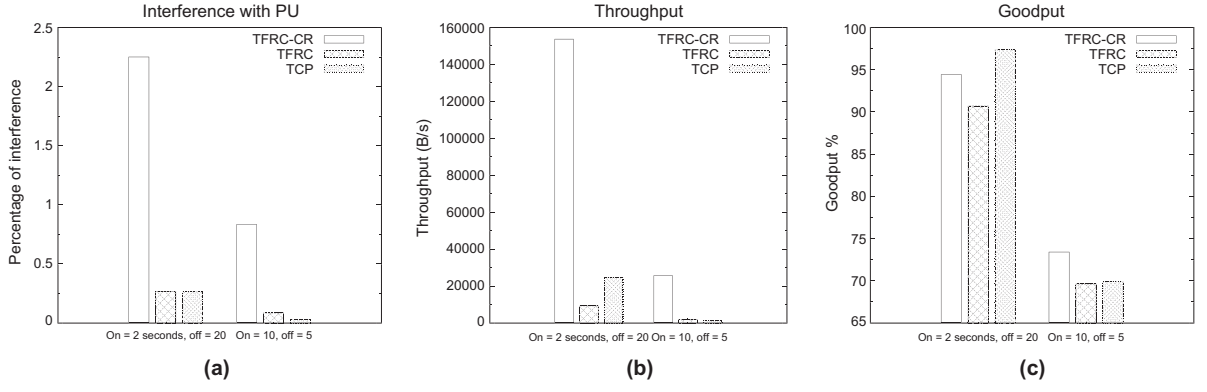


Fig. 12. The interference, throughput and goodput comparison are provided in (a), (b), and (c), respectively.

Please cite this article in press as: A.K. Al-Ali, K. Chowdhury, TFRC-CR: An equation-based transport protocol for cognitive radio networks, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.04.007

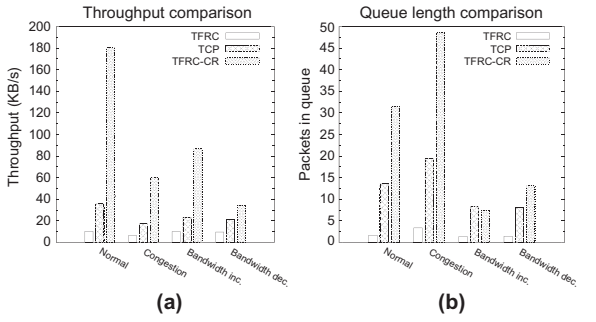


Fig. 13. Throughput and queue length (for node 3) comparisons under different stress events: none, congestion, bandwidth increase and bandwidth decrease.

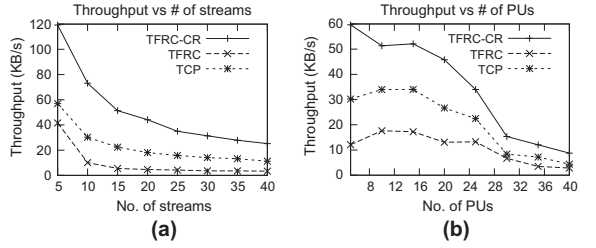


Fig. 14. Throughput as the number of streams and PUs change, respectively.

and PU environment. The on and off times for the PUs is set as an exponential distribution with averages 10 and 5 s, respectively. The x and y coordinates of the PUs are selected at random, and so is the source and destination node for each data streams. We study the impact of the number of active streams in Fig. 14a, with the number of PUs fixed at 15. In the second study, the number of streams is fixed at 10 in Fig. 14b, while the number of PUs is progressively increased. The rest of the parameters are identical to those discussed in the beginning of Section 6. In Fig. 14a and b, we can see that TFRC-CR provides higher throughput even as the density of the nodes and PUs increase.

7. Conclusion

We presented an equation-based transport protocol, TFRC-CR, which is geared to meet the demands of CR networks and presents a fundamentally different control mechanism compared to the typically used TCP-based schemes. TFRC-CR was demonstrated to perform significantly better than its classical counterparts TFRC and TCP with respect to both PU protection and transmission efficiency in a dynamically changing spectrum environment. Our protocol does not assume any cross-layer feedback or input from intermediate nodes, which aligns it with the traditional end-to-end paradigm in the evolving space of transport layer research for multihop CR networks.

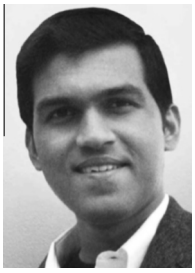
Acknowledgment

The authors thank Qatar University for the PhD. scholarship that is awarded to Abdulla Al-Ali. This material is based upon work supported by the US National Science Foundation under Grant No. CNS-1265166.

References

[1] I.F. Akyildiz, W.Y. Lee, K.R. Chowdhury, CRAHNs: cognitive radio ad hoc networks, in: Elsevier Ad Hoc Networks, vol. 7(5), July 2009.
[2] K.R. Chowdhury, M. Di Felice, I.F. Akyildiz, TP-CRAHN: a transport protocol for cognitive radio ad-hoc networks, in: IEEE INFOCOM 2009, April 2009.
[3] S. Floyd, M. Handley, J. Padhye, J. Widmer, Equation-based congestion control for unicast applications, in: Proc. of ACM SIGCOMM, August 2000.
[4] K. Sundaresan, V. Anantharaman, H.Y. Hsieh, R. Sivakumar, ATP: a reliable transport protocol for ad hoc networks, in: IEEE Trans. on Mob. Comput., vol. 4 (6), November–December 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 mechanism for mobile environments, in: Proc. IEEE INFOCOM, March 2000.
[7] P. Sinha, T. Nandagopal, N. Venkitaraman, R. Sivakumar, V. Bharghavan, WTCP: a reliable transport protocol for wireless wide-area networks, in: Wireless Networks, vol. 8 (2–3), March 2002.
[8] G. Holland, N.H. Vaidya, Analysis of TCP performance over mobile ad hoc networks, in: Wireless Networks, vol. 8 (2–3), March 2002.
[9] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, TCP selective acknowledgment options, RFC 2018 (October 1996).
[10] Y.B. Ko, N. Vaidya, Location-aided routing (LAR) in mobile ad hoc networks, in: Wireless Networks, vol. 6 (4), September 2000.
[11] W.Y. Lee, I.F. Akyildiz, Optimal spectrum sensing framework for cognitive radio networks, in: IEEE Trans. on Wireless Comm., vol. 7 (10), October 2008.
[12] V. Jacobson, Congestion avoidance and control, in: SIGCOMM Comput. Commun. Rev., vol. 18 (4), August 1988.
[13] M. Di Felice, K.R. Chowdhury, W. Kim, A. Kassler, L. Bononi, End-to-end protocols for cognitive radio networks: an evaluation study, in: Elsevier Performance Evaluation, vol. 68 (9), September 2011.
[14] A.O. Bicen, O.B. Akan, Reliability and congestion control in cognitive radio sensor networks, in: Elsevier Ad Hoc Networks, vol. 9 (7), September 2011.
[15] D. Sarkar, H. Narayan, Transport layer protocols for cognitive networks, in: Proc. of IEEE INFOCOM on Comp. Commun. Workshops, March 2010.
[16] H. Xiao, K.C. Chua, J.A. Malcolm, Y. Zhang, Theoretical analysis of TCP throughput in adhoc wireless networks, in: Proc. of IEEE GLOBECOM, December 2005.
[17] A.M.R. Slingerland, P. Pawelczak, R.V. Prasad, A. Lo, R. Hekmat, Performance of transport control protocol over dynamic spectrum access links, in: Proc. of IEEE DySPAN, April 2007.
[18] C. Luo, F.R. Yu, H. Ji, V. Leung, Optimal channel access for TCP performance improvement in cognitive radio networks, in: Springer Wireless Networks, vol. 1 (2), February 2011.
[19] C. Luo, F.R. Yu, H. Ji, V. Leung, Cross-layer design for tcp performance improvement in cognitive radio networks, in: IEEE Trans. on Vehicular Technology, vol. 59 (5), June 2010.
[20] A. Kumar, K.G. Shin, Managing TCP connections in dynamic spectrum access based wireless LANs, in: Proc. IEEE SECON, June 2010.
[21] X. Wang, X. Sun, C. Zhao, Z. Zhou, TCP-CReno – TCP enhancement using cross-layer for cognitive radio networks, in: Proc. AIAI, October 2010.
[22] F. Amjad, C. Zou, B. Aslam, Transparent cross-layer solutions for throughput boost in cognitive radio networks, in: Proc. IEEE CCNC, January 2013.
[23] N. Bapat, V.R. Syrotiuk, Adapting sensing and transmission times to improve throughput in cognitive radios ad hoc networks, in: IEEE WoWMoM, June 2012.
[24] <https://github.com/abdulla-alali/TFRC-CR/tree/CRAHN>.

Abdulla K. Al-Ali graduated with BS in Computer and Electrical Engineering from University of Miami, FL, USA. He received his MS in Computer Software Engineering from Northeastern University, Boston, MA, USA. He is currently a PhD candidate in Computer Engineering at Northeastern University. His research interests include transport layers specific to cognitive radio, developing and expanding modules in ns-2 and wireless datastream optimization in Android. He is a recipient of the Qatar University PhD scholarship.



Kaushik R. Chowdhury is Assistant Professor in the Electrical and Computer Engineering Department at Northeastern University, Boston, MA. He graduated with BE in Electronics Engineering with distinction from VJTI, Mumbai University, India, in 2003. He received his MS in Computer Science from the University of Cincinnati, OH, in 2006, and PhD from the Georgia Institute of Technology, Atlanta, GA in 2009. His MS thesis was given the outstanding thesis award jointly by the ECE and CS departments at the University of Cincinnati. He won the best paper award in the Ad Hoc and Sensor Networks symposium at the IEEE ICC conference in 2009, and currently serves on the editorial board of the Elsevier Ad Hoc Networks and Elsevier Computer Communications journals. His expertise and research interests lie in wireless cognitive radio ad hoc networks, energy harvesting, and multimedia communication over sensors networks. He is a member of the IEEE.

Please cite this article in press as: A.K. Al-Ali, K. Chowdhury, TFRC-CR: An equation-based transport protocol for cognitive radio networks, Ad Hoc Netw. (2013), http://dx.doi.org/10.1016/j.adhoc.2013.04.007