TCP with feed-forward source coding for wireless downlink networks

Jung Ryu, Sandeep Bhadra and Sanjay Shakkottai

***Abstract*—It is well-known that TCP connections perform poorly over wireless links due to channel fading. To combat this, techniques have been proposed where channel quality feedback is sent to the source, and the source utilizes coding techniques to adapt to the channel state. However, the round-trip time-scales quite often are mismatched to the channel-change time-scale, thus rendering these techniques to be ineffective in this regime. In this paper, we propose a source coding technique that when combined with a queueing strategy at the wireless router, eliminates the need for channel quality feedback to the source. We show that either in a multi-path environment (e.g., the mobile is multi-homed to different wireless networks) or in the presence of multiple TCP connections sharing the same wireless spectrum (where bandwidth can be opportunistically shared between different mobile users), the proposed scheme enables statistical multiplexing of resources, and thus increases TCP throughput dramatically.**

I. Introduction

TCP was designed and optimized with the assumption that the networks that it was supposed to operate over have highly reliable node-to-node links such that dropped packets due to bad links are highly unlikely. It was this trait of the wired network that TCP utilized to build a congestion control mechanism; a dropped packet is interpreted by TCP as a buffer overflow due to a congestion somewhere in the network.

However, a typical wireless link is designed with BER on the order of 10^{-5}, which translates into a packet drop probability of 5-10% for a 1KB packet; the drop probability can be much worse if the wireless channel is in deep, severe fade. If plain TCP is used over the wireless links without any modifications, this considerably reduces the average congestion window size and prevents it from enlarging to any significant portion of the ideal size, the bandwidth-delay product, resulting in a low utilization rate [15], [17], [3].

This paper addresses the problem of low TCP throughput in the simple topology of TCP senders connected via wireline network to intermediate wireless routers and TCP receivers connected by a wireless channel to one or more of these intermediate routers (see figure 1). An example scenario would be a cellular access network (such as UMTS/WiMax) where the cellular base station is connected to the wired backbone, and only the link between the base station and the mobile user is wireless. In addition, although multi-homing is not currently implemented in UMTS networks, with the introduction of femto-cells, it is conceivable that in a campus scenario with a number of femto-cells, the mobile user may be able to receive downlink data simultaneously on multiple links from multiple femto-cells; this motivates the multi-path model in figure 1.

A. State of the art: a matter of time-scales

To combat the adverse nature of the wireless network, multiple solutions have been proposed, all involving a separation of time-scales between the rate of channel variation and the TCP congestion window evolution. One can break the TCP connection between a wired server and a mobile into two components: wired and wireless [4]. However, this approach needs a proxy at the wireless base-station, and breaks TCP end-to-end semantics.

By contrast, one could protect TCP (without proxying at the wireless router) from channel-level variations by suitable physical layer schemes. Of these, the commonly deployed solution in UMTS/WiMax systems involves channel coding, adaptive modulation and/or automatic repeat request (ARQ or hybrid ARQ) deployed in a lower layer protocol to deal with packet drops resulting from channel variations that are at a much faster rate than the end-to-end TCP round-trip time (RTT). However, these schemes could lead to variations in the rate provided to the TCP connections, and can lead to sub-optimal TCP performance [9]. Papers such as [12] improve TCP performance over downlink wireless networks through dynamically adjusting PHY layer parameters optimized for TCP. However, such strategies require measurements both at the transport layer like TCP sending rate as well as physical layer information like channel quality.

The alternate solution (see [6], [26], [25]) is to code the data stream at a specific forward error coding rate at the application layer so that the decoded TCP data stream can withstand drops due to bad wireless channels. In [26], the authors use Reed-Solomon coding at a fixed rate to encode a stream of TCP packets in order to deal with random losses. In [25], the authors use network coding combined with an ACK scheme found in [24] and TCP-Vegas like throughput measurements to adapt TCP over wireless links. However, such an approach requires the variation in channel drop rate to be quasi-static relative to the time-scale of feeding back this channel drop rate information to the source so that the coding rate can be adjusted.

B. Motivation

1) Channel variation and RTT have same time-scale: In reality, however, there exist scenarios where the packet drop rate can change at the time scale of round-trip time of the TCP connection. For example, consider a mobile user

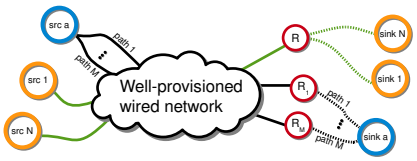


Fig. 1. N TCP-RLC connections (green) made over a well-provisioned wired network and a wireless router (red) and one connection (black) made over M paths. One can exploit multi-flow or multi-path diversity to increase throughput.

traveling at 2-5km/h using the current UMTS network (carrier frequency $\approx$ 2GHz in the U.S.) to download a file stored at a distant end of the Internet. This user's wireless channel coherence time is roughly around 20-50ms [21], a number well within the range of RTT for the Internet. (Coherence time is roughly a measure of how long a wireless channel stays constant, and therefore a rough measure of how fast the packet drop rate changes.)

In such scenarios, multiple ARQ requests and link-level ACKs and NACKs are unhelpful – they cause retransmission delays and timeouts that may adversely affect the RTT estimation and the retransmission time-out (RTO) mechanism, and therefore throughput. Moreover, forward error correction coding at a fixed rate (at the TCP source) is not helpful since the drop rate at the wireless link is not quasi-static relative to the feedback time scale. If the drop rate changes every RTT, the information about the drop rate will not reach the TCP sender in time to be useful since by the time the information reaches the sender, the drop rate would have changed. Thus, this mobile user's wireless downlink channel would be useless to track from the perspective of improving his TCP throughput; the channel quality feedback reaches the source too late, and is useless by the time the source gets it.

In summary, coding at fixed rate will not work when the wireless channel variation and RTT have the same time-scale.

2) Multiple path statistical multiplexing: There has been much research into multipath TCP connections. The obvious advantage of multipath TCP is that it can balance the load on the multiple paths such that paths experiencing temporarily high capacity can carry more packets than paths experiencing low capacity. Further, *multiple TCP connections can be useful for load balancing among multiple wireless interfaces* – for instance, in a situation where a mobile node is connected to a 3G network base-stations as well as a femto-cell base-station. In this scenario, one would want to get *statistical multiplexing* gain among the two wireless interfaces, as it is likely that the wireless fading state between the two interfaces will differ (e.g., when the 3G interface has a "bad" channel, the femto-cell interface could have a "good" channel). However to exploit this, a naive implementation would require that packets stored at the 3G base-station be transferred to the other base-station (femto) through a wired back-haul. Clearly, time-scales of RTT over the back-haul and channel variation would render this impractical.

A second issue one faces when running a TCP connection over multiple paths is the problem of out-of-order packet delivery, which can cause congestion window collapse even if the network has plenty of capacity [28]. [20] gets around this problem by delaying and reordering received packets before they are passed up to the TCP layer on the receive side. [29] uses duplicate selective ACKs (DSACK) and dynamically changes the duplicate ACK threshold to address the out-of-order problem.

By using random linear coding, our proposed TCP modifications can be naturally extended to multiple paths and we will show that coding + TCP enables the network to behave as though packets are "virtually shared" among the different base-stations without the need for a back-haul between the various base-stations. This in-turn leads to multiplexing gains. Further, coding + TCP can easily deal with out-of-order delivery of packets.

C. Other related work

Network coding: Recently, inspired by [1], [16] and others, network coding schemes have been used in the context of wireless networks in order to improve throughput. [10] and [23] use network coding at intermediate nodes and exploit the shared wireless spectrum to improve TCP throughput. In our approach, we use random linear coding (RLC) [18] at the end nodes to improve TCP throughput, and the intermediate router does not perform any coding operations. The concept of using random linear coding for TCP over wired networks has appeared recently [8]; however, our work here is for hybrid network with the goal of improving TCP throughput over time-varying wireless channels.

TCP window statistics under AQM: Lastly, we note that there exists a considerable body of literature [14], [27] on modeling the TCP window process in the presence of active queue management (AQM) systems, especially random early detection (RED) [11]. [27] presents a weak limit of the window size process by proving a weak convergence of triangular arrays; some of the mathematical model and treatment in our paper is based on their work. [2] presents a fluid limit of the TCP window process, as the number of concurrent flows sharing a link goes to infinity, and the authors show that the deterministic limiting system provides a good approximation for the average queue size and total throughput.

None of the previous works mentioned above treats the situation when the loss rate can not be tracked due mismatch between the channel change time-scale and the RTT time-scale. Our work shows throughput gains that can be achieved by exploiting multi-user or multi-path diversity when TCP is combined with an ACK scheme similar to the one in [25] and priority queueing strategy found in [7], plus RLC [18].

D. Main contributions

In this work, we employ (i) rateless coding, (ii) priority-based queueing at wireless routers and (iii) multi-path routing to demonstrate that throughput can be increased significantly for TCP over downlink wireless networks even when channel variations are on the same time-scale as RTT. In fact, *the key*

theoretical result underlying our proposed architecture shows full statistical multiplexing gain from multi-path TCP. Even in single-path routing TCP throughput in wireless access networks can still be increased due to statistical multiplexing gain from multiple flows sharing the same wireless router.

Specifically, our theoretical treatment shows that we can achieve TCP throughput of $\Theta\left(p_{\min}C\right)$, $\Theta\left(\mathbb{E}\left[1/P\right]^{-1}C\right)$, and $\Theta\left(\mathbb{E}\left[P\right]C\right)$ in single flow, multiple flow and multiple path cases, respectively, in the absence of channel quality feedback from the destination to the sender. Here, $p_{\min}$, $\mathbb{E}\left[1/P\right]$ and $\mathbb{E}\left[P\right]$ are the worst-case, inverse mean, mean probability of successful packet transmission for the time-varying wireless channel; C is the capacity of the wireless router. This in turn implies that the statistical multiplexing gain between (coding + multi-path TCP) and (multiple-single-path TCP) is of the order of $\Theta\left(\mathbb{E}\left[P\right]/p_{min}\right)$, where there are many path available with roughly comparable statistics.

Further, our modifications to TCP present an orderwise gain over the performance of plain TCP, which is $\Theta(1)$, in the presence of random packet loss for wired-wireless hybrid networks, where the random packet loss rates change at the RTT time scale and cannot be tracked.

II. Analytical Model & Results

A. Overview

Our modified TCP protocol, dubbed TCP-RLC, operates via performing random linear coding (RLC) over a block of data packets; the block size is equal to the TCP congestion window size, and both the coded and data packets are transmitted by the router at the wired-wireless network boundary for downlink transmission. When the receiver gets these data and coded packets, the receiver attempts to reconstruct the lost data packets from the successfully received data and coded packets. Before transmission, the sender marks a packet either high priority or low priority. The number of low priority packets transmitted is some fixed multiple, r, of the number of high priority packets transmitted; the number of high priority packets transmitted in any round-trip time is equal to the congestion window size maintained by the source. We refer to r as the redundancy rate.

The wireless router uses a priority transmission rule where high priority packets are transmitted before any low priority packets. In this paper, we assume that the wired network has enough spare capacity to handle the additional low priority packets. This assumption is a typical feature of the hybrid wired/wireless network, as the Internet is over-provisioned with excess capacity to stay ahead of the current traffic demand and to anticipate for the future increase. In addition, the wireless link is typically the bottleneck in the hybrid network architecture because wireless spectrum is expensive and is used near full capacity to maximize profit for the network operators.

We will describe the model necessary for our analysis and present our analytical results first. We will then give a brief description of the protocol itself in other subsections.

B. Network topology

We consider two network topologies. The first topology we consider is a set of $N \geq 1$ TCP connections made through a single wireless router R. The connections are made over links where only but the last link to the destinations is wired. We assume that the wireless router R is the point of congestion. The router R receives packets on its wireline interface and transmits packets across a noisy wireless channel to destination D_i, $i = 1, ..., N$. We will consider slotted time, with each unit time-slot corresponding to an RTT-interval. For the purpose of analysis, all connections made through R have the same RTT. In each RTT-interval the wireless router can transmit NC packets across the wireless channels, where C is the nominal capacity per RTT slot per connection of the wireless channel corresponding to the modulation and channel coding used. In other words, if the wireless channel experiences no error, then NC packets can be transmitted successfully from the wireless router in an RTT slot. For the purpose of analysis, we assume that the wireless router has no buffer space for storing packets from one RTT slot to another; the router's buffer is limited to what is needed for packet processing only. However, in the simulations, we relax this and allow buffers for storage (roughly one RTT worth of packets).

The second network topology we consider in this paper is a TCP connection made over $M \geq 1$ paths going through router R_1,..., R_M, each with capacity C (black connection in figure 1). As before well-provisioned wired network is assumed with the paths having the same RTT, and the wireless routers do not store packets from one RTT slot to another in the analytical model.

C. Wireless downlink channel

We model packet drops in the wireless channel between a wireless router and a TCP destination as a simple i.i.d. packet drop process whose parameter remains constant for each *RTT-interval*. This is similar to the block-noise model common in wireless communication literature. We index RTT intervals with t. Within each RTT-interval t, the probability that a packet transmitted over the air for destination i (or path i, in the multi-path context) is successfully received is given by $p_i(t) \in \{p_1 = p_{\min},\ p_2,\ \ldots,\ p_{|\Pi|} = p_{\max}\}$. In each time slot, $\mathbb{P}(p_i(t) = p_k) = \tilde{p}_k$. Thus, the channel packet-delivery-probability parameter itself changes with time (this corresponds to changing fading state over time), and at any time the actual packet delivery probability depends on the instantaneous value of this (random, time-varying) parameter.

D. TCP window dynamics

The TCP source i maintains a congestion window of size $W_i(t)$ for the t-th RTT interval. The congestion window is in units of packets; packets are assumed to be of fixed size. In each RTT slot, source i wants $W_i(t)$ data packets to be

transferred to sink i. We model the additive increase, multiplicative decrease (AIMD) evolution of the TCP congestion window as follows:

$$W_i(t+1) = \mathbf{1}_{\text{success}} \min\{W_i(t)+1, W_{\max}\} + \mathbf{1}_{\text{drop}} \lceil W_i(t)/2 \rceil$$

where the random variable $\mathbf{1}_{\text{success}} \triangleq 1$ when all data packets transmitted in the t-th RTT interval for destination i have been successfully recovered at the destination; $\mathbf{1}_{\text{drop}} \triangleq 1 - \mathbf{1}_{\text{success}}$ takes the value 1 when either (i) the receiver cannot recover one or more data packets corrupted by the packet drop process or (ii) a packet is marked by the router due to the presence of an *active queue manager* (AQM). $W_{\max}$ represents the limitation placed on the congestion window size due to limited buffer at the TCP receiver. We will neglect TCP timeouts in our analysis to simplify our model and analysis.

E. Random linear coding

We will assume that source i takes $W_i(t)$ data packets and generates $rW_i(t)$ coded packets in t-th RTT slot as follows: let each packet $x_{ik}, k = 1, 2, \ldots, W_i(t)$ be represented as an element of some finite field $\mathbb{F}_q$; choose elements $\alpha_{ikj} \in \mathbb{F}_q$ uniformly at random and generate a coded packet $y_{ij} = \sum_{k=1}^{W_i(t)} \alpha_{ikj} x_{ik}$ for $j = 1, 2, \ldots, rW_i(t)$. Here, $r > (1 - p_{\min})/p_{\min}$ so that $p_{min}(W(t) + rW(t)) > W(t)$. The receiver can decode, with very high probability, any dropped data packets if sufficient number of innovative coded and data packets are received, as the field size from which the coding coefficients are drawn increases.

Accordingly, in the rest of the work, we will make the following assumption as a simplification.

Assumption 1: Suppose W data packets are used to generate coded packets via RLC. If G coded packets are received by the TCP destination, then upto G missing data packets out of the W data packets can be recovered.

Hence, if the number of missing data packets from W exceeds G in an RTT slot, the congestion window will halve. For detailed exposition on RLC and justification of assumption 1, see [19].

F. Priority transmission and multiple paths

When multiple paths are used, the source TCP-RLC protocol is made of two types of controllers. The first is the source controller; this is where the congestion window and hence coding block size $W(t)$ is maintained. The source controller then passes the data and coded packets to the paths controllers.

Each path controller maintains a path congestion window $w_l(t)$ for path l. Before a packet is transmitted on path l in RTT slot t, it is appended with a block number $b_l(t)$ and the block size that equals $w_l(t)$. The evolution of $w_l(t)$ is similar to $W(t)$: $w_l(t+1) = \mathbf{1}_{\text{success}} \min\{w_l(t)+1, W_{\max}\} + \mathbf{1}_{\text{fail}} \lceil w_l(t)/2 \rceil$ where $1_{\text{success}} = 1$ if $w_l(t)$ packets are successfully received in RTT slot t; $1_{\text{fail}} = 1 - 1_{\text{success}}$.

Before the packets are sent out on a path, they are marked either high or low priority. The number of high priority packets sent out in any given RTT slot is equal to the path congestion window $w_l(t)$. For each high priority packet sent out, the path controller obtains r packets from the service queues and marks them low priority and sends them out.

When a single path is used, the source controller is the lone path controller and $W(t) = w_1(t)$; in addition, data and coded packets are marked high and low priority, respectively.

G. Wireless router

A priority rule is implemented at the router R to handle the streams of high and low priority packets – we assume that the high priority packets are transmitted first by the router R. For example, in the single flow model ($N = 1, M = 1$), the remainder of the nominal channel capacity C, $C - W(t)$, is used to transmit low priority packets in t-th RTT slot, where $W(t)$ is the number of high priority packets transmitted by the source. Further, we assume that at any RTT interval a sufficient number of auxiliary low priority packets are always available at the router R. Thus, the wireline link $S_i - R$ is considered to be over-provisioned to accommodate sufficient number of low priority packets for the worst channel error parameter. See the overview section II-A for our justification of this assumption.

We also make the assumption that the router R maintains a pair of queues for each TCP connection made through that router, i.e. R maintains $2 \times N$ queues. While such a number that scales with the number of flows would be prohibitive for routers in the Internet core, we argue that it is reasonable for wireless downlink routers, as these routers serve relatively small number of mobile users in the same cell. In addition, per flow queue maintenance is already done in cellular architectures for reasons of scheduling, etc. (see [5]).

Although we assumed no buffer in the analytical mode, we allow one FIFO queue for high priority packets and one LIFO queue for low priority packets for each flow. The capacity of these buffers has no bearing on the performance as long as the LIFO queue can hold one RTT worth of packets and FIFO queue is large enough for packet processing and can smooth out jitters in delay.

H. Analytical results

The proofs to the theorems in this section are in [22]; however, we provide plausible explanations to convince the readers for the single flow and multi-path cases. For the purpose of analysis, we assume that the wireless router has the ability to mark packets either 1 or 0 according to some probabilistic function f (see [11] for a similar marking mechanism). That is, a packet is marked 1 with probability f; f depends on such parameters as C, the sum of window sizes (multiple flow case), etc. For simulations and practical implementations, this assumption is not needed. When the sink receives a packet marked 1, it alerts the source to reduce the congestion window size via a message piggy backed on

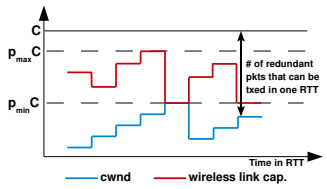


Fig. 2. Evolution of the congestion window for single flow and for each path in multi path. Congestion window size varies from $p_{\min}C$ to $0.5p_{\min}C$; however in multipath, packets are encoded across paths, so that $p(t,i)C$ number of packets received by the sink on path i can be summed up with packets arriving on other paths, for the total of $\sum_{i=1}^{M} p(t,i)C$ to increase throughput.

an ACK packet. Note that probabilistic marking related to f is different from priority marking.

1) Single flow: When we have a single TCP-RLC connection through a wireless router with capacity C, we have the following results; we use $\mathbb{E}[W(t)]$ to denote the mean window size. For technical reasons, the single-flow analysis requires that $p_1 > 0.5$.

Theorem 1: There exists a marking function such that $\mathbb{E}[W(t)] \geq 0.75p_1C - 1$ as $C \to \infty$. ∎

2) Qualitative explanation - single flow: The reason we can only guarantee such bound is that as soon as the congestion window size increases beyond p_1C, the wireless channel may become bad temporarily to be unable to support p_1C packets per RTT. For example, suppose $W(t) = p_1C+1$; then, only $C-p_1C-1$ redundant packets will be transmitted by the router. If the wireless channel success probability is p_1 in RTT slot t and assuming exactly p_1C packets are successfully received by the sink, then the number of successfully received packets will be less than $W(t)$ and the congestion window will be halved in RTT slot $t+1$. (See figure 2.) Due to our assumption that the wireless channel conditions change every RTT, this temporary bad condition happen frequently enough as $C \to \infty$ to prevent $W(t)$ from enlarging it beyond p_1C.

3) Multiple paths: When the source is able to manage M multiple paths, we have the following result that says we can achieve (normalized per-path) throughput of the order $\Theta(\mathbb{E}[P]C)$ for large M. We assume that all paths have the same channel characteristic, i.e. $p_n(t) \in \{p_1 = p_{\min}, ..., p_{|\Pi|} = p_{\max}\}$ with $\mathbb{P}(p_n(t) = p_k) = \tilde{p}_k$ for all paths $n = 1, ..., M$ and for all t.

Theorem 2: For any $\rho > 1$ and M. Then,

$$\mathbb{E}[W(t)] \geq \max\Big\{0.75p_1MC/\rho^2 - 1, \min\Big\{\frac{0.75\mathbb{E}[P]MC}{\rho^2}\left[1 - 3\delta_1 - 2\left(e^{-1}+\delta_2\right)\right], \beta'\left[1 - 3\lfloor\beta'\rfloor^{-1} - 2\left(e^{-1}+\delta_2\right)\right]\Big\}\Big\}$$

where $\delta_1, \delta_2 \to 0$ as $C \to \infty$ and where

$$\beta' \in [p_1MC/\rho^2, \mathbb{E}[P]MC/\rho^2]$$

is the solution to the equality $1/\beta' = \exp(-Ml'(1-\rho\beta'/MC))$ (for M, C large enough, we can show a solution exists) and where

$$l'(a) = \max_{-\infty<\Theta<\infty}\{\Theta a - \log M'(\Theta)\} \quad (1)$$

and $M'(\Theta) = \mathbb{E}\left[e^{\Theta(1-P)}\right]$. ∎

4) Qualitative explanation - multiple paths: The path congestion window evolves the same way as in single flow case, varying between p_1C and $0.5p_1C$. (See figure 2.) However, in each RTT slot, the number of packets successfully received by the sink on path i is then $p(t,i)C$, and summing over all paths, we get $\sum_{i=1}^{M} p(t,i)C \approx \mathbb{E}[P]MC$. If the packets are coded across paths, we can have the normalized congestion window $W(t)/M$ reach $\mathbb{E}[P]C$. However, to reach this number, we might need a large number of paths. However, we show in simulations, we can increase throughput dramatically even with 2–3 paths in certain situations.

Note that coding across paths is what allows us to increase throughput. If we were to encode packets on a given path separately from packets on other paths (i.e. having M separate single path TCP connections), the throughput we would achieve is of the order $\Theta(Mp_{\min}C)$, whereas the statistical multiplexing gain due to coding across paths results in a throughput of $\Theta(M\mathbb{E}[P]C)$, thus resulting in an improvement factor of $\mathbb{E}[P]/p_{min}$ when there are many path available with roughly comparable statistics.

5) Multiple flows: When we have N TCP-RLC flows going through a single wireless router, and when the router has the channel information so that it may control the number of low priority packets for each flow based on channel information, we have the following result that says we can achieve throughput of the order $\Theta\left(\mathbb{E}[1/P]^{-1}C\right)$. We assume that all flows have the same channel statistic; $p_i(t) \in \{p_1 = p_{\min}, ..., p_{|\Pi|} = p_{\max}\}$ with $\mathbb{P}(p_i(t) = p_k) = \tilde{p}_k$ for all flows $i = 1, ..., N$ and for all t.

Theorem 3: There exists constants $\delta > 0$ and ρ and a suitable marking function such that for any $\alpha > 1$,

$$\mathbb{E}[W(t)] \geq \left[\alpha^{-1}\mathbb{E}[1/P]^{-1}(C - \log(C/2)/\rho - 1)\right] \times (1 - 2e^{-1} - 2\delta)$$

as $C \to \infty$. ∎

Note that the bound is intuitive when you consider that the sum of all capacity demands in RTT slot t, $\sum_{i=1}^{N} W_i(t)/p_i(t)$, must be less than or equal to NC. (In order for $W(t)$ data packets to be transferred to the sink over that wireless channel whose packet delivery probability is $p(t)$, the wireless router has to transmit roughly total of $W(t)/p(t)$ packets (data+coded).) If the router intelligently controls the number of redundant packets transmitted over the air for each flow based on the channel quality information $p_i(t)$, we can get close to order $\mathbb{E}[1/P]^{-1}C$ per flow. Since redundant (low priority) packets are stored in LIFO queues, determining which redundant packet to transmit is simple – transmit the head LIFO packet as this packet is the most relevant one to the current RTT time slot. Lastly, one can show that order $\mathbb{E}[1/P]^{-1}C$ is the throughput one

can achieve if the channel quality information is instantly available at the source.

III. TCP-RLC Protocol

We break the description of our proposed protocol into three subsections.

A. Source architecture

1) ACK and pseudo ACK: TCP-RLC uses two types of ACK's: ACK, as used in the plain TCP and pseudo ACK, which we describe here. Plain ACK's cumulatively acknowledge the reception of all packets with frame numbers smaller than or equal to the ACK.

In our context, a lost data packet can be "made up" by a future coded packet, and we would like the sliding congestion window to slide forward and have delay-bandwidth product worth of packets in transit. Thus, we use a strategy similar to that in [25], where degrees of freedom are ACKed. In our context, we refer to this as a pseudo ACK, which simply ACKs any out of order data packet or coded packet that helps in decoding the smallest-index missing packet (e.g., if the sink has received packets 1, 2, 3, and 7, the smallest-index missing packet is 4). Note that with regular TCP, out-of-order packets would trigger duplicates ACK's that would lead to a loss of throughput.

2) Sliding congestion window: The source maintains a congestion window W, which is the same as the size of the coding block for TCP-RLC. All packets transmitted are marked either high or low priority; the source allows only W high priority packets to be in transit. Each time a high priority packet is transmitted, it transmits r low priority packets as well.

The source maintains variables, last-ACK and SN. All packets with frame numbers lower than last-ACK are assumed to have been successfully received. SN is the frame number of the starting packet in the coding block currently being transmitted. If pseudo ACK or ACK arrives "acknowledging" the reception of SN, a new coding block is encoded and readied for transmission, with the coding block size being $W + 1$ packets. This is because acknowledgment of packet number SN implies a RTT has been elapsed, which is enough time for W packets to have been successfully received and decoded by the sink.

3) Multiple paths: When multiple paths are used, the marking of packets is done by the paths independently. Each path is maintained by a path controller and the controller maintains a congestion window, $cwnd_i$; another top-level controller maintains $cwnd$, which is used as the size of the coding block. After the packets are encoded, they are passed to path controllers (thus, coded packets are "mixed" across paths, which in-turn leads to statistical multiplexing across paths). Each path marks packets either high or low priority. In each RTT slot, the number of high priority packets in transit is equal to $cwnd_i$. The number of low priority packets (coded packets) per path is equal to $r \times cwnd_i$. Each packet going out on a path contains the block number and block size, which is also equal to $cwnd_i$. In each RTT slot, if the sink on path i receives $cwnd_i$ packets (either high or low priority), $cwnd_i$ increases by one; else $cwnd_i$ reduces by half. Note that single path is just a special case of multiple paths; and the variables $cwnd$ and $cwnd_1$ are the same. (In the single-path case, the role played by the separate path controllers is subsumed into the source controller.)

There are two levels of ACK's; one level for source controller (source level ACK, consisting of ACK and p-ACK) and the other for path controllers (path level ACK). Note that: *(i)* path level ACK is a new ACK introduced for multi-path TCP – there is no equivalent in the single-path case, and *(ii)* the three types of ACKs described here is abstracted into a single indicator function for success/drop in the analysis (see Section II-D). Source controller level ACK's affect $cwnd$ and moves the coding window; path controller level ACK's affect $cwnd_i$'s and moves the block numbers.

B. Destination architecture

Upon reception of a packet, the destination examines if it is the next expected (i.e., smallest-index missing) packet. If it is, the destination sends an ACK cumulatively acknowledging all packets upto and including the just received packet. If not, the destination sees if the packet is an innovative packet that can be used to decode the next expected. (A packet is innovative if it is linearly independent of all packets received so far, i.e. it can help in decoding the next expected packet. For a complete definition of innovative packets, see [19] and [13].) In case the packet is helpful, the destination sends a pseudo ACK with next expected packet number + total number of innovative packets accumulated that can help in decoding the next expected packet. If the packet does not help in decoding the next expected packet, the destination sends a duplicate ACK.

C. Illustrative example

We illustrate the additive increase, multiplicative decrease component of TCP-RLC and the pseudo ACK's using three examples. The examples are for when a TCP connection uses a single path, and the data and coded packets are marked high and low priority, respectively. Although the TCP source uses retransmission time-outs for when there is no response from the sink for long period of time, we do not show this in our examples.

1) Additive increase: In figure 3(a), the packets P1-P4 are encoded together. P2-P4 are dropped due to bad wireless channel; however, the sink has received enough coded packets to recover them. As the sink receives these coded packets, it sends out a pseudo ACK for each one. This enables the source to move the congestion window forward, keeping the "pipe" between the source and the sink full.
2) Multiplicative decrease (bad wireless channel): In figure 3(b), too many packets are dropped due to bad channel to recover P1-P4. When packets P5-P7 arrive

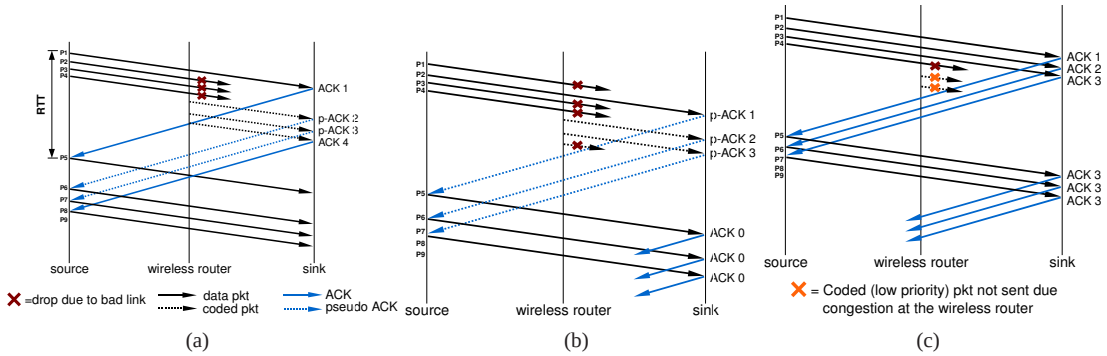


Fig. 3. Illustrative example

at the sink, duplicate ACK's are sent out and the source will cut the congestion window.

3) Multiplicative decrease (congestion at wireless router): In figure 3(c), the router is busy transmitting data (high priority) packets, and no or very few coded (low priority) packets will be transmitted. Hence, not enough packets are received to recover any lost packets in P1-P4.

IV. Simulation Results

We simulate three types of networks: 1) single flow with single path from the source to the destination, 2) single flow but with multiple paths, where we exploit path diversity, and 3) multiple flows going through one wireless router, where we exploit user diversity. In all our simulations, the random coefficients used to encode each packet are drawn from a field of size 8191; thus, for coding block size of smaller than ~ 500, the probability of two coded packets in the same coding block not being "independent" were negligible. We fixed the size of all packets to 256 bytes. Each flow had two buffers allocated at the router each of the size equal to the transmission rate times the RTT. We have used bimodal channel profile, $\Pi = \{(p_{\min}, \widetilde{p}_{\min}), (p_{\max}, \widetilde{p}_{\max})\}$. $p_{\max}$ is set to 1 in all our simulations. Each time the wireless channel changed, it stayed constant for some random time according to the uniform distribution with parameters 100ms to 200ms; on average, the channels stayed constant for 150ms. For the single flow cases (single path and multipath), we fixed the sum of all link delays so that the total link delay is 150ms. Time-out clock is set to expire after 3 $\times$measured RTT. RTT was measured using IIR filtering: measured RTT = 0.9 $\times$ old measured RTT + 0.1 $\times$ new measured RTT. The redundancy factor r was such that $(1+r)/p_{\min} \approx 2$, with r being an integer. We assume perfect uplink channel from the mobiles to the wireless routers for the end-to-end TCP ACK's.

A. Single path TCP flow

In this case, we varied $p_{\min}$ from 0.5 to 0.9 in increments of 0.1 and set $\widetilde{p}_{\min} = \widetilde{p}_{\max} = 0.5$. C was varied from 500Kbps to 2Mbps in the increments of 500Kbps. The average throughputs (total number of bits transferred/simulation time) we obtained are shown in figure 4(a) as a function of $p_{\min}$.

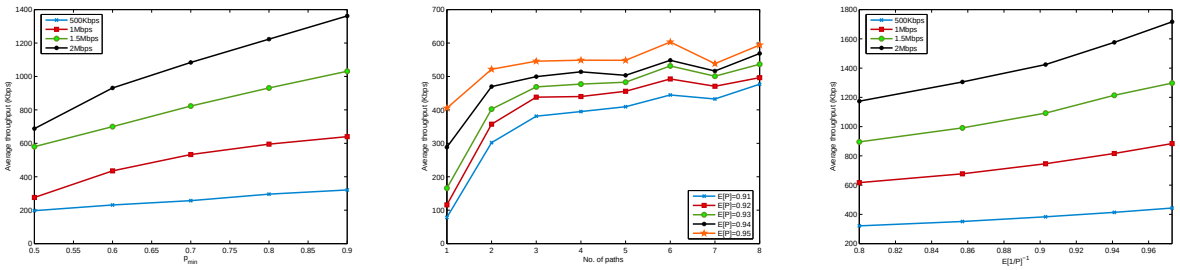
B. Multiple path

In our simulations, all paths have the same bimodal channel profile and have the same capacity. We varied p_1 from 0.1 to 0.5 in increments of 0.1 and set $\widetilde{p}_1 = 0.1$ and $\widetilde{p}_2 = 0.9$. Thus, $\mathbb{E}[P]$ varied from 0.91 to 0.95. This corresponds to the scenario where the downlink channel can be controlled to provide good capacity most of the time (90% of the time), but bad channel conditions can occur frequently enough to destroy TCP throughput (10% of the time). Note that using fixed coding rate adjusted for the average channel quality would not work for this scenario; coding is not needed most of the time, and is useless when needed. We varied the number of paths from 1 to 8. We set the total capacity to 1Mbps; so that per path capacity is 1Mbps/M, where M is the number of paths in our simulation.

The average throughputs we obtained are shown in figure 4(b) as a function of the number of paths. As the number of paths increases, the average throughput increases towards $\mathbb{E}[P]CM$ (CM is fixed) in a concave manner, indicating that going from one path to two paths gives much gain in throughput, especially when $p_{\min}$ is small. The throughput should reach 700Kbps (for $\mathbb{E}[P] = 0.91$) to 730Kbps (for $\mathbb{E}[P] = 0.95$).

C. Multiple TCP flows

This scenario corresponds to the case where multiple TCP flows are sharing one bottleneck wireless router, with the bottleneck router being just one hop before the destinations. In this simulations, we have 30 flows, and we assume that the channel quality information (the packet delivery probability) is available to the router. The router then uses that information to control how many coded packets are sent for each flow. For this simulations, we used p_1 varying from 0.5 to 0.9 in increments of 0.1 and set $\widetilde{p}_1 = 0.25$ and $\widetilde{p}_2 = 0.75$. Flows had different total link delays; the link delays were picked randomly from a uniform distribution with parameters 100ms to 200ms and fixed for the duration of the simulation. The average throughputs we obtained are shown in figure 4(c). The simulated throughput is between 70–80% of the expected throughput.



(a) Average throughput vs. $p_{\min}$ for single path TCP-RLC

(b) Average throughput vs. no. of paths for multiple path TCP-RLC connection

(c) Average throughput vs. $\mathbb{E}\left(1/P\right)^{-1}$ for multi-flow TCP-RLC

Fig. 4. Simulation results

V. Conclusion

We proposed modifications to the TCP controller to adapt it to the hybrid downlink networks. The throughputs obtained via our modifications were shown to be proportional to $\mathbb{E}\left[1/P\right]^{-1}$ and $\mathbb{E}\left[P\right]$ in multi-flow and multi-path scenarios, respectively, without the source tracking the channel quality information. Further, our analysis shows a statistical multiplexing gain between (coding + multi-path TCP) and (multiple-single-path TCP) of the order of $\Theta\left(\mathbb{E}\left[P\right]/p_{min}\right)$, where there are many path available with roughly comparable statistics.

VI. Acknowledgement

This research was supported by a grant from the US Department of Defense, NSF grants 0519401, 0347400, and the DARPA ITMANET program.

References

[1] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46:1204–1216, 2000.

[2] F. Baccelli, D. McDonald, and J. Reynier. A mean-field model for multiple TCP connections through a buffer implementing RED. *Performance Evaluation*, 49(1-4):77–97, September 2002.

[3] H. Balakrishnan, V. Padmanabhan, and R. Katz. The effects of asymmetry on TCP performance. In *Proc. ACM/IEEE Mobicom*, pages 77–89, September 1997.

[4] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE/ACM Transactions on Networking*, 5(6):756–769, 1997.

[5] P. Bender, P. Black, M. Grob, R. Padovani, N. Sindhushayana, and A. Viterbi. CDMA/HDR: A bandwidth efficient high speed wireless data service for nomadic users. *IEEE Communications Magazine*, pages 70–77, July 2000.

[6] L. Benyuan, D. Goeckel, and D. Towsley. TCP-cognizant adaptive forward error correction in wireless networks. In *Globecom '02*, 2002.

[7] S. Bhadra and S. Shakkottai. Looking at large networks: Coding vs. queuing. In *Proceedings of IEEE Infocom*, Barcelona, Spain, 2006.

[8] T. Bonald, M. Feuillet, and A. Proutiere. Is the "Law of the Jungle" sustainable for the internet? In *Proceedings of INFOCOM*, 2009.

[9] E. Chaponniere, S. Kandukuri, and W. Hamdy. Effect of physical layer bandwidth variation on TCP performance in cdma2000. In *Vehicular Technology Conference*, 2003.

[10] P. S. David and A. Kumar. Network coding for TCP throughput enhancement over a multi-hop wireless network. In *Communication Systems Software and Middleware and Workshop*, 2008.

[11] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

[12] M. Ghaderi, A. Sridharan, H. Zhang, D. Towsley, and R. Cruz. TCP-aware resource allocation in CDMA networks. In *Proceedings of ACM Mobicom*, 2006.

[13] T. Ho and D. Lun. *Network Coding: An Introduction*. Cambridge University Press, 2008.

[14] C. Hollot, V. Misra, D. Towsley, and W. Gong. On designing improved controllers for AQM routers supporting TCP flows. In *Proceedings of IEEE Infocom*, 2001.

[15] Y. Itan, K. Xu, and N. Ansari. TCP in wireless environments: Problems and solutions. *IEEE Radio Communications*, pages 27–32, March 2005.

[16] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft. XORs in the air: Practical wireless network coding. In *ACM SIGCOMM*, 2006.

[17] T. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth delay products and random losses. *IEEE/ACM Transactions on Networking*, 5(3):336–350, June 1997.

[18] M. Luby. LT codes. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002.

[19] D. S. Lun. *Efficient Operation of Coded Packet Networks*. PhD thesis, Massachusetts Institute of Technology, 2000.

[20] B. Radunović, C. Gkantsidis, D. Gunawardena, and P. Key. Horizon: Balancing TCP over multiple paths in wireless mesh network. In *Proceedings of ACM Mobicom*, 2008.

[21] T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, 2002.

[22] J. Ryu, S. Bhadra, and S. Shakkottai. TCP with feed-forward source coding for wireless downlink networks. WNCG Technical Report, UT Austin, 2009.

[23] L. Scalia, F. Soldo, and M. Gerla. PiggyCode: a MAC layer network coding scheme to improve TCP performance over wireless networks. In *Wireless on Demand Network Systems and Services*, January 2008.

[24] J. K. Sundararajan, D. Shah, and M. Médard. ARQ for network coding. In *Proceedings of ISIT*, Toronto, Canada, 2008.

[25] J. K. Sundararajan, D. Shah, M. Médard, M. Mitzenmacher, and J. Barros. Network coding meets TCP. In *Proceedings of IEEE Infocom*, 2009.

[26] O. Tickoo, V. Subramanian, S. Kalyanaraman, and K. K. Ramakrishnan. LT-TCP: End-to-end framework to improve TCP performance over networks with lossy channels. In *Proceedings of IWQoS*, June 2000.

[27] P. Tinnakornsrisuphap and A. Makowski. On the behavior of ECN/RED gateways under a large number of TCP flows: Limit theorems. *Queuing Systems*, 2006.

[28] Z. Ye, S. V. Krishnamurthy, and S. K. Tripathi. Effects of multipath routing on TCP performance in ad hoc networks. In *IEEE Globecom*, 2004.

[29] M. Zhang, B. Karp, S. Floyd, and L. Peterson. RR-TCP: a reordering-robust TCP with DSACK. In *IEEE ICNP*, 2003.