Abstract—The poor performance of TCP in wireless networks is further diminished in IEEE 802.11-based ad hoc networks due to the unnecessary actions of on-demand routing protocols when interference-induced packet drops occur at the medium access control (MAC) layer. In this paper, we present a cross-layer hop-by-hop congestion control scheme designed to improve TCP performance. The proposed scheme attempts to determine the actual cause of a packet loss and then coordinates the appropriate congestion-control response among the MAC, network, and transport protocols. The congestion control efforts are invoke at all intermediate and source nodes along the upstream paths directed from the wireless link experiencing the congestion-induced packet drop. Simulation results show that the proposed scheme achieves up to 70% higher throughput than TCP-Reno.

I. INTRODUCTION

In this work, we consider the problem of congestion control in large-scale mobile ad hoc networks. In current packet switched networks (i.e. the Internet), congestion control schemes are incorporated into the Transmission Control Protocol (TCP), which provides reliable connection-oriented service to upper layers of the TCP/IP protocol suite. Recent research has shown that TCP suffers significant performance degradation in ad hoc networks mainly because TCP’s loss-based congestion control scheme and additive increase multiplicative decrease (AIMD) window mechanism are not suitable for the unique characteristics of ad hoc networks, such as multihop wireless connections, shared wireless medium, scarce capacity and instability of routes due to mobility [1]–[5]. In short, the aggressive nature in which the TCP AIMD scheme consumes bandwidth quickly overshoots shared channel capacity, resulting high collision rates and MAC-layer packet losses.

Moreover, recent work by Nahm et al. showed that the link layer packet drops driven by TCP mechanisms also trigger over-reactions of on-demand routing protocol, causing additional TCP performance impairments [6]. A node failing to deliver a packet through a wireless link regards the link as broken without identifying the real cause of the packet drop. Such a link breakage activates the propagations of route error messages along the reverse paths of routes directed to the broken link. Then the traffic sources that lose routes have to start route discovery by flooding route request messages into the networks. The unnecessary route maintenance and discovery operations result in the waste of limited network capacity, insufficient usage of queues and increased end-to-end transport packet delivery latency.

In this paper, we introduce a cross-layer hop-by-hop congestion control scheme to improve TCP performance in multihop wireless networks by solving the aforementioned false routing disruption problems and providing the on-demand routing protocol with congestion control capability. Basically, the proposed scheme enable nodes to intelligently differentiate link layer packet drops due to congestion or link breakage, so as to prevent unnecessary routing reactions to packet drops.

As opposed to many proposed schemes that address on solving TCP congestion control problem in multihop wireless networks from end-to-end point of view, our scheme is novice in that it enables the on-demand routing protocol to be congestion-aware and actively participate in congestion control. Moreover, our scheme can be deployed together with legacy TCP as it contains minimized modications on TCP, thus it guarantees TCP compatibility. The simulation results show that our scheme improves TCP throughput by up to 70%.

The rest of this paper is organized as follows. In Section II, we discuss related work. In Section III, we present the details of our cross-layer hop-by-hop congestion control design. Section IV evaluates the performance of proposed scheme. The conclusion is given in Section V.

II. RELATED WORK

Many of the previously proposed schemes for addressing the problems of TCP, include TCP with adaptive pacing (TCP-AP) [7], the transport protocol (ATP) [8] and rate-based transport control (RBCC) [9], belong to the category of end-to-end congestion control since the transport source needs feedback information from the transport receiver to decide congestion control actions, i.e. set the congestion window size or transmission rate. An end-to-end congestion control scheme message experience propagation delays along reverse path. Alternatively, our proposed hop-by-hop congestion control
scheme has the advantages of minimizing congestion notification delay and being able to delay transmission of outstanding data packets by performing congestion control on a hop-by-hop basis.

ADTCP identifies network states such as congestion, route failure and so on by analyzing metrics extracted from end-to-end TCP packets [10]. TCP-DOOR infers changes of a route by detecting out-of-order TCP packets [11]. In TCP-ELFN [1], the TCP sender is notified directly of a packet loss caused by link failure via an intermediate node. These scheme improve TCP performance by avoiding reduction of the congestion window when packet losses are due to route changes. However, as already discussed previously, the routing protocol generally treats each packet delivery failure (whether due actual route failure or MAC-layer congestion) link breakage event and activates routing maintenance and rediscovery procedures, causing the TCP senders to obtain false information on network states.

The mostly closely related work to the scheme proposed in this work is TCP-FeW [6]. Our scheme is targeted on averting performance loss from false routing reactions driven by TCP window mechanism, which differs from TCP-FeW proposed in [6]. TCP-FeW reduces unnecessary routing operations by slowing TCP window growth rate, instead of directly reacting to congestion-induced route failure. Our scheme identifies link packet drops due to signal interference by monitoring signal reception power, hence directly preventing unnecessary routing maintenance and re-discovery when packet drops due to signal interference happen. Moreover, our proposed scheme does cause significant changes to the standard TCP protocol and interoperable with existing TCP and can be applied transparently to UDP.

III. CROSS-LAYER DESIGN OF HOP-BY-HOP CONGESTION CONTROL

In this section, we present the details of our cross-layer hop-by-hop congestion control scheme, which coordinates actions among MAC, network and transport layers to improve overall end-to-end performance. Specifically, it prevents unnecessary route discovery operations when MAC packet drops are due to contention by identifying causes of these drops. It allows the on-demand routing protocol to temporally freeze transmissions through the congested link, and to disseminate congestion warning to all affected traffic sources. The TCP senders can immediately reduce transmission rate after receiving congestion warning directly from network layer.

A. Identifying false link failures

Generally, a packet delivery failure at a wireless link is due to one of following causes: wireless channel error, interference of nearby transmission signals, or a broken wireless link because two communicating nodes moves out of each other’s transmission range. Since IEEE 802.11 requires the receiver to acknowledge the data packet and retries transmission for several times if not receiving the ACK, the probability of packet loss due to wireless channel error is small enough to be omitted. Therefore, it is reasonable for us to assume that a link packet drop is either caused by broken link or due to signal interference.

The relation between signal reception power $P_r$ and the distance $d$ between the transmitter and the receiver can be approximated as:

$$P_r = P_0 d^{-\alpha}$$

where $P_0$ is the reception power at a reference distance (usually one meter) from the transmitter [12]. The value of $\alpha$, called the distance-power gradient, varies with the surrounding terrain conditions such as buildings and street layouts. Typically $\alpha$ has a value between 2 and 4. This equation shows that the signal reception power decreases exponentially as $d$ increases. Other factors that affect the values of the signal reception power include the transmitter and the receiver antenna gains. Since terrain and antenna gains normally do not change in a small time scale like several seconds, and a node usually transmits at a constant transmission power unless a power control scheme is applied, consecutive drops of the signal reception power in a short period can only be attributed to an increase in the distance $d$. Two communicating nodes are said to be out of each other’s transmission range, when the reception power at the receiver drops below the reception
Algorithm 1 Inferring causes of packet losses

Variables:
\( L_t \): The latest signal reception power in decibel
\( T \): A predefined window
\( t \): Duration from the last signal reception to till now
\( \tau \): Duration between receptions of last two signals
\( L_r \): Exponential average of reception power in \( T 
\( \bar{\delta} \): Exponential average of reception power slope
\( L_c \): The estimated reception power at current time
\( L_{thr} \): The reception power threshold in decibel

When receiving a new signal reception power sample

\[
\text{if } \tau < T \text{ then }
\delta = 0.5 \frac{(L_l - L_r)}{L_r} + 0.5 \bar{\delta} \\

\bar{L}_r = L_t + \delta T \\
\text{else}
\delta = \frac{(L_l - L_r)}{L_r} \\
\bar{L}_r = L_t \\
\text{end if}
\]

When detecting a MAC packet drop

\( \bar{L}_c = \bar{L}_r + \delta t \)

\text{if } \bar{L}_c < L_{thr} \text{ then }
\text{Report link failure}
\text{else}
\text{Report signal interference}
\text{end if}

sensitivity \( p_{sen} \), which is the lowest acceptable reception power for a node to decipher data from a received signal. We conducted simulation experiments in the network simulator Qualnet [13] to validate the above reasoning. A 4-hop FTP connection is set as illustrated in Fig. 1(a). Other detailed simulation configurations are specified in Section IV. While the FTP session is in progress, node 3 moves out of range of node 2 and node 4. During this process, the signal reception power from node 3 to node 2 is recorded. Fig 1(b) illustrates the decreasing trend of the reception power as the distance between node 2 and node 3 increases.

This finding forms the basis of the algorithm to infer whether a wireless link is broken when it experiences a packet drop. Theoretically, if a node fails to deliver a packet to one of its neighbors and the most recent signal reception power from the neighbor is below \( P_{sen} \), then link failure can be accounted for the packet loss. However, there exists an interval from when the latest signal reception power is recorded to when the packet loss happens. If the interval is fairly large, the latest signal reception power recorded is outdated for estimating current distance to the receiver, thus leads to wrong inference. To solve the problem, we propose to estimate current signal reception power according to historical reception power data.

We assume that the relative moving speed \( v \) between the sender and the receiver is constant during a small window \( T \), which typically ranges from several hundred milliseconds to several seconds, depending on the value of \( v \). The idea is that the samples of signal reception power measured in the most recent interval \( T \) are considered valid. Let \( \bar{P}_c \) be the estimation of the current signal reception power, and \( P_l \) be the latest recorded signal reception power, the relationship between \( \bar{P}_c \) and \( P_l \) can be given by

\[
\frac{\bar{P}_c}{P_l} = (\frac{d_l + vt}{d_l})^{-\alpha}
\]

(2)

Where \( d_l \) is the distance when the latest sample is recorded, and \( t \) is the duration since the latest sample. When power is represented in decibel, 2 is expressed as

\[
\bar{L}_c - L_l = -10\alpha(\log_{10}^{(d_l + vt)} - \log_{10}^{d_l})
\]

(3)

Since \( t < T \), within a small interval \( t \), \( 10\log_{10}^{(d_l + vt)} - 10\log_{10}^{d_l} \) can be considered approximately as \( \beta(d_l + vt - d_l) = \beta vt \), so

\[
\bar{L}_c - L_l = -\alpha \beta vt
\]

(4)

Since \( v \) is considered constant in \( T \), the current reception power in decibel can be estimated by

\[
\bar{L}_c = L_l + \delta t
\]

(5)

where, \( \delta L \) is the slope for the signal reception power in decibel during window \( T \). In actual computation, we use the exponential average of signal reception power \( \bar{L}_r \) and exponential average of signal reception power slope \( \delta \) to estimate current reception power. Each time a new signal reception power is recorded, \( \delta \) and \( \bar{L}_r \) are updated sequentially

\[
\delta = \begin{cases} 
0.5 \frac{(L_l - L_r)}{L_r} + 0.5 \bar{\delta}, & \text{if } \tau < T; \\
\frac{(L_l - L_r)}{L_r}, & \text{otherwise}
\end{cases}
\]

(6)

\[
\bar{L}_r = \begin{cases} 
L_t + \frac{T - \tau}{T} \bar{L}_r, & \text{if } \tau < T; \\
L_t, & \text{otherwise}
\end{cases}
\]

(7)

Where \( \tau \) is the interval between the most recent two signal reception power samples. When a packet drop happens, the current signal reception power in decibel is estimated as

\[
\bar{L}_c = \bar{L}_r + \bar{\delta} t
\]

(8)

Thus if \( \bar{L}_c \) is below signal reception sensitivity in decibel \( L_{sen} \), then MAC layer will report a link failure to the routing protocol. Nevertheless, since the signal bit error rate is mainly determined by the signal-to-noise ratio, when the signal reception power is close to \( P_{sen} \), both a small signal interference and a temporarily increased environment noise can fail the packet delivery. We consider the wireless connectivity is not reliable in this situation and it is better to trig a link failure event for the routing protocol to find a more reliable route. Therefore, the proposed algorithm defines a signal reception threshold \( P_{thr} \) larger than \( P_{sen} \), so that a link failure event is reported if the estimated current signal reception power is below such threshold when a packet loss happens. The complete inference process is illustrated in Alg. 1.

The accuracy of the proposed inference algorithm is tested through simulation. The simulation scenario is configured as follows. Thirty mobile nodes are uniformly distributed in a
square terrain. The mobility model is the random waypoint model with pause time of 30s and speed ranging from 0-10m/s. Four ftp and four CBR traffic sources are randomly started during simulation. Table I compares the total inferred and actual numbers of packet losses due to link failures or signal interference. Simulation results show that the inference accuracy is to 96.4% and 86.1% respectively when there exits no fading or Rayleigh fading in propagation channel.

B. Hop-by-hop congestion control at network layer

If a node fails to deliver a data packet through a wireless link that is not subject to broken, as defined in the previous section, the node infers that the packet drop is due to signal interference from nearby transmission. Once a wireless link experiences a MAC packet loss due to signal interference, it can be inferred that the contention area in which the link resides is congested, hence a hop-by-hop congestion control scheme is activated. At network layer, the proposed scheme relies on the dissemination of congestion control messages by the on-demand routing protocol. It also requires the routing protocol to modify routing tables to allow delayed transmission through the congested area. We choose the implement the scheme by modifying AODV at network layer. The detailed operations of proposed scheme in AODV are described as follows.

1) At the node experiencing a packet drop due to signal interference: When a node fails to deliver a packet through a wireless link due to signal interference, MAC reports to the routing protocol that a congestion-induced packet drop happens at the link. AODV then marks all the route entries forwarding packets through the link as congested in the routing table. These congested routes will be forbidden to forward packet temporarily to relieve congestion. In order to do so, each congested route entry is also associated with a frozen timer, whose expiration time points to the timestamp when the route entry can be used again. How long should a congested route be frozen? The route frozen time must be at least large enough to allow these packets to move out of the interference range of the receiver. With our simulation setup, we choose the frozen time value as 40ms plus some random jitter values within 10ms, which is used to avoid potential collisions when multiple overlapping routes are recovered simultaneously.

After labeling congested route entries, the node preserves the backlogged packets to be forwarded along these routes. Once these routes are re-enabled, the node will attempt to deliver these salvaged packets immediately.

Finally, the node generates a route congestion message that contains information about those route entries marked as congested, including their frozen time. The route congestion message is transmitted to potential affected neighbors via broadcasting.

2) At the node receiving a route congestion message: When a node receives a route congestion message, it obtains the congested routes stored in the message. Then it searches over its own routing table for route entries which match those in the congested route message. If a route entry has the same destination as the one of congested routes, and the next hop field of this entry points to the node that sent the route error message, then this route entry is marked as congested and is disabled for similar time period as the matching route. This strategy ensures that once the upstream nodes receives a route congestion message, it will cease forwarding packets to the link experiencing the packet drop due to congestion, throttling upstream transmission even before transport sources detect congestion.

Afterwards, the current node checks if it has neighbors that forwarding packets to the congested link through itself. If so, it generates and broadcasts a new route congestion message.

3) At the node sending a packet over the congested route: When a node needs to send a packet over a congested route, the packet may be a packet originating from the transport layer of the node, or an IP packet from another node. If it is the first case, the node will buffer this packet and send a message to the transport layer to notify that the path is congested. Otherwise, the node will buffer the data packet as well as generate and send another route congestion message.

C. TCP’s reaction to congestion notifications

After receiving the congestion notification messages from the IP layer, the TCP senders reduces its congestion window in half to decrease transmission rate.

D. Discussions

Our proposed cross-layer hop-by-hop congestion control scheme has a number of benefits. By identifying the causes of MAC layer packet drops, it avoids unnecessary route
error propagations and route re-discovery procedures triggered by false link broken events. Allowing the routing protocol to temporarily freeze routes directed to the congested area helps to relieve congestion immediately, while an end-to-end congestion control scheme experiences considerable delay before a traffic source can detect congestion in its path. Our hop-by-hop congestion control has the advantage of being able to immediately freeze data transmission through the congested. Using AODV, route congestion messages are propagated along the reverse paths of all active routes using the congested link. Subsequently, all traffic sources using the congested link will be notified to quench their transmission, regardless of whether they detect a packet loss in their own flows. An example shown in Fig. 3-D explicitly demonstrates such advantages. There are four TCP flows from sources S1, S2, S3 and S4 to destinations D1, D2, D3 and D4, sharing the same wireless link from i to j. When there is a contention-induced packet drop at link i – j, route congestion messages are transmitted backwards to all these TCP sources. The proposed scheme also makes effort to salvage queued data packets to increase packet delivery ratio. The only control overhead required by this scheme is the propagation of route congestion messages, which have the same size and propagation routes as the normal AODV route error messages. That is, the proposed scheme does not generate any additional overhead since, the same number and size of route error messages will be generated for a single MAC packet loss event.

IV. PERFORMANCE EVALUATION AND COMPARISONS

In this section, we evaluate the proposed scheme through simulations over various scenarios in chain and grid topologies. Since the objective of the proposed cross-layer congestion control scheme is to reduce TCP throughput loss due to unnecessary route maintenance operations over multihop wireless networks, we choose TCP end-to-end throughput as the performance index. We also run the same set of simulations with TCP enabled with pre-configured static routes and TCP-FeW. With pre-configured static routes, TCP flows do not experience frequent route failures and need not to activate route re-discovery process, so it provides an indication of the performance upperbound of the proposed scheme. TCP-FeW is proposed to limit TCP’s aggressiveness so as to prevent the over-reaction of on-demand routing protocols. It serves as a reference scheme to which we compare our cross-layer hop-by-hop congestion control scheme. Unless explicitly specified, simulations are configured using the same setting described in Section III.

A. Simulation Environment

We implement our congestion control scheme and conduct simulations in network simulator Qualnet 3.7. Unless explicitly specified, all simulations are configured with following settings. At the physical layer, the propagation pathloss model is two-ray ground. IEEE 802.11b is used as physical layer protocol with the transmission power fixed as 10dbm and reception sensitivity value as -89dbm. A combination of these settings provides an approximate radio range of 300 meters. The physical channel bandwidth is 2 Mbps. MAC layer protocol is IEEE 802.11 DCF with a RTS/CTS hand-shake required before the data packet is transmitted. At network layer, predefined static routing or AODV is configured as the routing protocol. At transport layer, TCP-Reno is the selected TCP variant with delayed-ACK option enabled. The default TCP maximum segment size (MSS) used is 512 bytes. In all simulation topologies, each pair of neighboring nodes have a distance of 250 meters.

B. Chain topology

Four schemes are tested over the 3- to 30-hop chain topologies with a single FTP session is established for each scenario. The tested schemes include TCP-Reno configured with AODV routing protocol, TCP-Reno with predefined static routes, TCP-FeW with AODV routing protocol and TCP-Reno enabled with proposed cross-layer hop-by-hop congestion control scheme. For the convenience of discussion, we name TCP with the proposed scheme as TCP-chc. In our simulation setting, TCP-FeW increases the congestion window size by a fraction of 0.1 of the maximum segment size for each round trip time. We tested various window increment fraction values over a number of scenarios, TCP-FeW with window increment fraction value of 0.1 obtains the highest throughput. The simulation results are presented in Fig. 3.

Fig. 3(a) compares the average end-to-end FTP throughput obtained with different schemes. We observe that TCP-chc obtains a 20%- 70% higher throughput than TCP-Reno configured with AODV does, and its performance is very close to that of TCP-Reno configured with static routes. Compared with TCP-FeW, it obtains higher throughput for TCP connections of larger hop counts (larger than 12). Overall, both scheme achieves similar throughput enhancement over TCP-Reno. Fig. 3(b) and Fig. 3(c) compares average congestion window size and retransmission ratio between the four schemes. Retransmission rate is calculated by the TCP sender as the number of retransmitted packets divided by the total transmitted packets, so it can be regarded as an approximate metric as packet loss rate. Since TCP-FeW adopts a less aggressive window increment scheme, it has the smallest the average congestion window size and lowest packet retransmission ratio. The other three schemes have the similar average congestion window sizes, but both TCP-Reno with static routes and TCP-cue reduced the packet retransmission ratio by avoiding dropping queued packets due to false route failures.

We also ran simulations over a 10×10 grid topology with each pair of neighboring nodes having a distance of 250 meters. Using the grid topology, two overlapping, two cross FTP flows, and two parallel FTP flows with equal or different hop counts have been tested separately. All these six mixed flows have been also run simultaneously as a representative scenario for more congested situations. We compare performance of TCP-chc to that of TCP-Reno and TCP-FeW with dynamic routing and the results are illustrated in Fig 4. Each
TCP−FeW and TCP-chc significantly enhance TCP throughput in multihop wireless networks which coordinates the congestion responses across the transport, network, and transport layer protocols and (2) we use simulations to compare the proposed scheme to legacy TCP and TCP-FeW over several scenarios. Simulation results show that the proposed scheme obtains significant throughput improvement over legacy TCP, and has achieved higher performance than TCP-FeW in some scenarios.

V. CONCLUSIONS AND FUTURE WORK

The lack of coordination between MAC and on-demand routing protocols results in frequent over-reactions of routing protocols, which further impairs end-to-end transport performance. In this paper, (1) we propose a cross-layer hop-by-hop congestion control scheme to improve TCP performance in multihop wireless networks which coordinates the congestion responses across the transport, network, and transport layer protocols and (2) we use simulations to compare the proposed scheme to legacy TCP and TCP-FeW over several scenarios. Simulation results show that the proposed scheme obtains significant throughput improvement over legacy TCP, and has achieved higher performance than TCP-FeW in some scenarios.

REFERENCES


