Use of Congestion-Aware Routing to Spatially Separate TCP Connections in Wireless Ad Hoc Networks

Zhenqiang Ye
Dept. of Electrical Engineering
University of California, Riverside
Riverside, CA 92521
Email: zye@cs.ucr.edu

Srikanth V. Krishnamurthy
Dept. of Computer Science & Engineering
University of California, Riverside
Riverside, CA 92521
Email: krish@cs.ucr.edu

Satish K. Tripathi
Dept. of Computer Science & Engineering
University at Buffalo, SUNY
Buffalo, NY 14260
Email: tripathi@buffalo.edu

***Abstract*— TCP sessions in ad hoc networks compete with each other for bandwidth. The use of shortest path routing can result in multiple TCP sessions being channeled via a few congested areas or hotspots. As a consequence, most of these multiple TCP sessions interfere with each other and hence, experience significant performance degradations. Spatially separating the TCP sessions such that they inflict much lower interference effects on each other may provide gains in performance. In this paper we first investigate the possibilities of achieving such gains by considering a centralized, ideal, and unrealistic congestion-aware routing approach. We find that spatial separation benefits are possible with the considered approach, and can especially help long (in terms of hop count) TCP connections. We then consider the implementation of a distributed routing protocol to achieve the aforementioned spatial separation benefits. We find that due to practicalities such as the need for the exchange of congestion state, the existence of stale congestion information and the creation of sub-optimal paths, the benefits due to spatial separation are considerably undermined. We perform both macroscopic simulations and microscopic studies of specific constructed examples to understand the reasons and quantify the various effects with both the centralized and the distributed approaches. Our studies suggest that achieving *noteworthy* performance gains by spatially separating TCP sessions may be extremely difficult if not impossible in ad hoc networks.**

Keywords: TCP, spatial separation, congestion-aware routing, ad hoc networks

I. Introduction

When multiple TCP sessions exert interference on each other, they end up losing packets and hence cause each other to throttle. This leads to significant performance degradations. In this paper, our objective is (a) to study whether there exist benefits due to spatial separation of TCP connections so as to reduce inter-connection interference and (b) to implement a congestion-aware routing approach that can exploit benefits due to spatial separation of TCP connections if any and thus, improve TCP performance.

Several congestion-aware routing (or load-aware routing) schemes have been proposed [1], [2], [3], [4]. The studies show that the congestion-aware routing protocols provide a lower end-to-end packet delay and lower packet dropping rate. However, most of the work use UDP connections and CBR traffic to evaluate the performance of the congestion-aware routing schemes. Instead of keeping a constant packet sending rate as in a UDP connection, a TCP connection always tries to maximize its throughput via congestion control and congestion avoidance mechanisms. The congestion control and congestion avoidance mechanisms detect packet losses and in response the TCP source shrinks its congestion window size. Therefore, TCP may exhibit different performance traits, as compared to UDP, with a congestion-aware routing protocol.

In this paper, our objective is to first examine if by spatially separating TCP connections by intelligently implementing routing functionalities, we can alleviate inter-connection interference effects and thereby improve performance. Toward this, we consider a centralized, ideal (but unrealistic) scenario, in which there are multiple TCP connections in a static ad hoc network. Each TCP connection is routed on a path such that the interference effects of a connection on other connections are minimized. The computation of the least congested path is based on the global awareness of the topology and the traffic distribution over the network (i.e., the paths on which the TCP connections are routed). We name the centralized routing scheme as the Centralized Congestion-Aware Routing (CCAR) scheme. The TCP performance with the CCAR scheme is compared with that of its performance in a practical scenario in which a shortest path routing protocol is used (in this paper we use AODV protocol [5]). We find that if the source-destination pair are close to each other (within 4 hops), there are no *spatial separation* benefits seen and the TCP goodputs[1] achieved by such connections are not enhanced by congestion-aware routing. This is a consequence of the fact that if congestion occurs near a short TCP connection, either the source or the destination is within the congestion area[2], or the connection has to find a much longer path (than the shortest one) in

This work was supported by DARPA under contract number: FTN F30602-01-2-0535.

[1]Here we define TCP *goodput* as the number of sequenced bits that a TCP receiver receives per second (duplicate packets are not counted).

[2]The congestion area cannot be bypassed, no matter which path the connection uses.

order to bypass a congestion area. Since the goodput of short TCP connections is sensitive to the length of a path, it is difficult for short TCP connections to obtain spatial separation benefits in either of the two cases. For longer connections, the performance of the CCAR scheme demonstrates that gains in goodput due to spatial separation of connections is possible. The ratio of the goodput achieved by the CCAR scheme to that achieved by shortest path routing increases with the distance between the source-destination pair.

Next, we design a Distributed Congestion-Aware Routing (DCAR) protocol. Our attempt is to mimic the behavior of the CCAR scheme to the extent feasible (without global awareness). In order to do so, it is required that each node possesses some state information, that characterizes the interference patterns in the node's close proximity. We modify the AODV protocol (with HELLO messages) to make it "congestion-aware". We find that several factors prevent the DCAR scheme from providing the same benefits that were observed with the CCAR scheme. These factors are (a) additional overhead incurred for exchanging congestion information among nodes, (b) stale congestion information, and (c) lack of global information at each node that inhibits the computation of optimal paths.

We also observe that the DCAR scheme helps long TCP connections. However, this is at the expense of hurting short TCP connections. In order to elucidate the behavioral traits that cause the above macroscopic effect, we consider a simpler constructed example for microscopic study. The example clearly demonstrates that, there is a competition for bandwidth among TCP flows in ad hoc networks. With the DCAR scheme, a longer connection can find longer paths that are affected to a less extent by other TCP flows than with shortest path routing. However, this is not to say that it is not affected nor that it does not exert interference on other TCP flows. By choosing the longer but less congested path we find that the sources of the longer TCP connections are more aggressive in injecting packets into the network. This, in turn, increases the congestion experienced by shorter TCP connections, that are unable to bypass congested regions effectively. This causes their goodputs to decrease.

The remainder of this paper is organized as follows. In Section II, we provide a brief description of related work on congestion-aware routing and improving TCP performance by means of modifications to the routing layer. Section III describes our simulation environment. We study our centralized, ideal routing approach in Section IV. In Section V we discuss the distributed congestion-aware routing scheme and evaluate its effects on TCP performance. Finally, we conclude the paper in Section VI.

II. Related Work

A limited number of congestion-aware routing schemes have been proposed in the literature. In [1], a Dynamic Load-Aware Routing (DLAR) was proposed. The number of packets in the interface queue of a node is used to quantify the traffic load at the node. In [2], a Load-Balanced Ad hoc Routing (LBAR) protocol was proposed. The traffic load experienced by a node is defined as the total number of routes passing through the node and its neighbors. A Load-Sensitive Routing (LSR) scheme was proposed in [3] and in this work the traffic load at a node is defined as the total number of packets being queued at the mobile node and at its neighbors. A Delay based Load Aware On-demand Routing scheme (D-LAOR) was proposed in [4]. D-LAOR tries to discover the shortest delay path by estimating the packet delay at each node. In the aforementioned previous work, only UDP connections and CBR traffic are considered in evaluating the performance of the congestion-aware routing schemes. Since UDP does not back-off or vary its transmission rate in response to failures, the effects of link failure and end-to-end packet delay are less dramatic on UDP performance. In comparison, TCP detects packet losses and adjusts its packet sending rate in response to packet losses to avoid congestion (by means of some congestion control and congestion avoidance mechanisms). Therefore, one might expect that TCP with congestion-aware routing performs different from UDP with congestion-aware routing. In particular, since TCP sources can more aggressively transmit packets to commensurate with the lower levels of congestion on a path that is discovered with congestion-aware routing, we expect that TCP may in fact gain much more. Different from previous work, in this paper we study the effects of the congestion-aware routing on TCP performance.

The COPAS protocol [6] uses node-disjoint paths for TCP-DATA packets in the forward direction and for TCP-ACK packets in the reverse direction to eliminate interaction between TCP-DATA and TCP-ACK packets of the same connection. In contrast to COPAS, [7] proposes the use of the same route for both TCP-DATA and TCP-ACK packets in order to reduce the total number of links that may be occupied by the connection. The effects of multi-path routing protocol on TCP performance have also been studied in [8], [9]. With multi-path routing multiple paths are computed from the source to the destination. The studies from [9] show that using multiple paths concurrently does not help in improving TCP performance and the authors propose using the shortest path as the primary route and the shortest-delay path as the backup. However, they do not discuss whether the multi-path routing scheme simply provides enhanced robustness to failures or if it does provide spatial separation between TCP flows.

In this work, in contrast with [9], we attempt to spatially separate TCP flows, and analyze via constructed examples, as well as, by means of macroscopic simulation studies, the reasons behind the observed behavioral traits of TCP with a congestion-aware routing policy.

III. The Simulation Environment

Since, our simulations form an integral part of our discussions on whether spatial separation can even help in improving TCP performance we first present our simulation framework.

For our studies we use the Network Simulator *ns-2* [10]. The Distributed Coordination Function (DCF) defined in the IEEE 802.11 standard [11] is used at the MAC layer.

The radio model is similar to a commercial radio interface, Lucent's WaveLAN, which is a shared-media radio with a nominal bit-rate of 2Mb/sec, a nominal communication range of 250 meters, and a sensing range of 550 meters[3]. We made modifications to the AODV [5] routing protocol to enable our distributed version of congestion-aware routing (to be described in Section V). The performance metric that we are interested in is the ratio of goodput that is observed with our proposed scheme to that observed with a traditional shortest path routing protocol.

In our simulations we place 200 nodes in a 2500m x 1000m region. All nodes are static and mobility is not considered in this paper[4]. In each simulation iteration, a random scenario is generated; 5 source-destination pairs are randomly chosen and TCP connections are established between these pairs. These TCP connections begin sequentially. The initiation instances of consecutive TCP sessions are separated by 1 second. Each TCP connection lasts for 100 seconds and is then terminated. The simulation results reported in this paper represent the average results over 1000 different scenarios.

TCP New-reno is used in all our simulations. The length of each TCP packet is 1460 bytes. In order to avoid inactivity due to a sequence of TCP back-off operations which are triggered by a series of packet losses due to link failures, in our simulations, the TCP sender disables its retransmission timer and enters a standby mode upon receiving a route error (RRER) packet. Subsequently, the sender sends out a packet periodically until an acknowledgment is received; the period is equal to the current value of its retransmission timer. Our approach is similar to the ones used in [12] and [13]. The maximum congestion window size is set to be 8 [13].

IV. Examining if Spatial Separation can Yield Gains: A Centralized Congestion-Aware Routing (CCAR) Approach

In this section, we consider a centralized ideal case to explore if there are benefits due to spatial separation. In the centralized ideal scenario, we assume that each node knows the topology of the entire network. We use a cost function similar to the one used in [2] to measure the level of congestion at each node. The cost function is the total number of TCP connections passing through the node and the nodes that are within its sensing range [5]. Specifically, each node knows every TCP source-destination pair in the network and the path on which each connection is routed at any given time. The congestion level of a node is then abstracted by a weight assigned to that node. For every TCP connection that goes through a node, or through one of its neighboring nodes that lie within its sensing range, the weight of the node is incremented by W_{TCP}.

$$W_{node_i} = n_i * W_{TCP} \ . \tag{1}$$

where, n_i is the number of TCP connections within node i's sensing range. The weight of the link between node i and node j (represented by $link_{i,j}$) is computed to be the maximum of the weights of the two nodes i and j and a link penalty denoted by W_p,

$$W_{link_{i,j}} = MAX(W_{node_i}, W_{node_j}) + W_p \ . \tag{2}$$

The link penalty, W_p, is used to take the effects of path length on TCP performance into consideration. In our congestion-aware routing scheme, even though, some longer paths may be less congested than the shorter ones, the extra links increase the round trip time experienced by packets and may cause an inefficient use of the wireless bandwidth. By imposing the link penalty W_p, we attempt to prevent TCP connections from choosing extremely long paths.

In our simulations (with this ideal case) we consider 5 TCP connections. Two heuristic strategies are used for choosing a path for a TCP connection. In heuristic strategy I, the connections are initiated sequentially. When a route is needed, the source node computes a minimum weight route (the sum of the weights of the links along the chosen route is the minimum computed from all the possible routes); the route is used until it breaks; when the route breaks, a new minimum weight route is computed. Pre-existing TCP connections do not change their paths when a new TCP connection is initiated. In heuristic strategy II, we assume that the 5 TCP connections begin simultaneously. We examine all possible permutations[6] of routing the 5 TCP connections in the network. For each permutation we estimate the overall congestion cost in the network. This is denoted by the sum of the weights of all the nodes for that permutation. We then assign routes to each TCP connection from the permutation that minimizes this overall congestion cost. During a simulation run, a TCP connection continues to use the assigned route in spite of false link failures that might occur. Note that in static networks, all the link failures are false link failures. A false link failure is caused by the fact that a sender may fail to obtain a CTS response from the receiver after several consecutive RTS attempts. The receiver fails to respond either because the medium at the receiver side is sensed to be busy or because the NAV (Network Allocation Vector) value of the receiver is not zero [11]. In order to make the comparisons with "*TCP with the shortest path routing*" fair, we initiate a route discovery process at each instance of a false route failure. Furthermore, a route is not computed until a route reply (RREP) reaches the route request (RREQ) originator. Note here that since link

[3]Nodes within the communication range can communicate with each other; nodes within the sensing range of a transmitting node can sense the transmission of the node, but the Signal-to-Noise (S/N) is too low to decode the signal.

[4]Note that our main objective here is to understand the implications of congestion and not mobility. The protocol will need additional overhead to cope with mobility. Since our results demonstrate that even in static scenarios, the overhead can render congestion-aware routing prohibitive, we can only expect the benefits of the congestion-aware routing to be further undermined with mobility.

[5]We count the TCP connections passing through the neighboring nodes that are within a node's sensing range due to the fact that, the transmission of these neighboring nodes can be sensed by the node, and thus preclude the node from accessing wireless medium. In short, the TCP connections passing through the node are affected by those connections that pass through these neighboring nodes.

[6]There are 5! such permutations.

failures are ignored, the sender uses the same singular route throughout the connection and never switches routes. Thus, the only purpose of the RREP and RREQ is to emulate traffic overhead.

In our simulations, we set $W_p = 1$ and $W_{TCP} = 1$[7]. Figure 1 shows that when two nodes are within four hops of each other, the ideal strategies provide no benefits as compared with TCP with shortest path routing. The reason for this observation is due to the fact that the transmission of a node can be sensed by all the nodes that are within its sensing range. Since the sensing range is more than two times larger than the communication range, if congestion occurs in the vicinity of such a short connection, typically either the source or the destination, or both, are within the affected area. In such cases, the bottlenecks cannot be avoided and thus no spatial diversity benefits are possible. When the source-destination pair are further apart, benefits are seen due to the spatial separation of TCP sessions. Furthermore, the two heuristic methods are similar in terms of the achieved goodput. We varied the value of W_p and found that $W_p = 1$ gives us the best performance. If the value of W_p is too large, W_p dominates the route discovery process and usually the shortest path has the least congested weight. In this case the congestion-aware routing scheme performs the same as the shortest path routing scheme. On the other hand, if the value of W_p is too small, the path with the least congestion weight is usually much longer than the shortest one. Since the goodput of short TCP connections is very sensitive to path length, the goodputs of such connections are decreased significantly with the use of extremely long paths.

Since short TCP connections might be expected to increase the TCP window more rapidly than long TCP connections (i.e., cause higher levels of congestion), we consider modifying W_{TCP} such that it is inversely proportional to the length of the connection in terms of hops. Thus, the presence of shorter TCP connections in the vicinity of a node, causes larger increases in the assigned weight W_{TCP}. In other words, for a given TCP connection, we set the increment in W_{TCP} due to the connection to be $1/L$, where L was the hop count of the particular connection. We observed that with this modification the heuristic strategies have a performance that was almost identical to that of TCP with the shortest path routing. The reason for this is that decreasing the value of W_{TCP} has the same effect as increasing the value of W_p. If W_{TCP} is set to be $1/L$, W_p dominates the route discovery process and the shortest path usually has the least congestion weight (as discussed earlier).

Finally, in heuristic strategy I, instead of choosing the minimum weight route, we consider choosing the route that had the *least congested bottleneck link* among the first K least congested edge-disjoint paths (K is set to 5 in our simulations[8]. In other words, if the maximum of the weights of the links on a particular route say $route_j$ was $W_{max,j}$, we choose the route R that had the minimum of this weight, i.e., route R had a link with maximum weight

$$W_{max,R} = MIN_j(W_{max,j}) \ .$$

In case, there were ties, among the tied routes, the route with the minimum total weight was chosen. The idea behind this strategy was to avoid bottleneck links that were *most* congested. The performance of this scheme is represented by Heuristic strategy I (modified) in Figure 1. We see that this strategy yields a performance that is almost identical to that of the other strategies. The reason for this observation lies in the fact that, the K least congested paths are usually closely coupled and the difference in the weights of their bottleneck links is insignificant. Generally, it is observed that the path with the least congested bottleneck link is the path with the least total congestion weight.

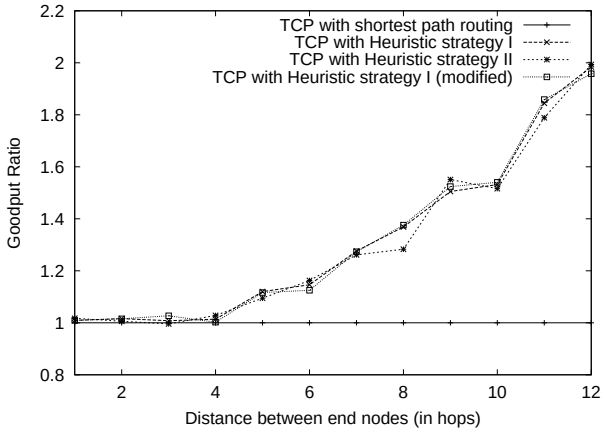

Fig. 1. Performance comparison of the heuristic strategies vs shortest path routing

The above simulation results seem to suggest that when a new TCP session is initiated, if one could somehow intelligently choose the least congested path, due to spatially separating the connection from congested areas, one could potentially see performance gains. However, due to inherent capacity limitations of the broadcast medium, the benefits are limited only to long TCP connections.

V. Distributed Congestion-Aware Routing (DCAR) Scheme

Our next objective is to investigate whether the benefits due to spatial separation seen with our CCAR scheme are possible via a distributed protocol. Thus, we implement a Distributed Congestion-Aware Routing (DCAR) scheme to mimic the CCAR strategy described in Section IV. In the DCAR scheme, each node i keeps track of the number of TCP

[7]Simulation results with other parameter values were worse in terms of the achieved goodput than for the case when $W_p = 1$ and $W_{TCP} = 1$; we omit these results from the paper due to space limitations.

[8]Larger values of K result in the discovery of much longer paths, which degrades TCP goodput. Smaller values of K may inhibit the identification and exclusion of congested bottleneck links.

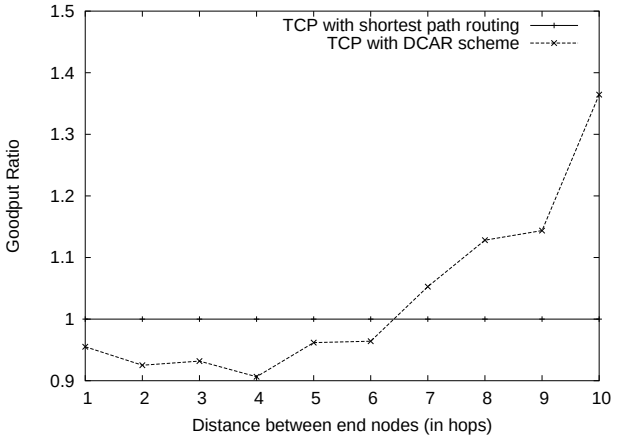

Fig. 2. Performance comparison of the DCAR scheme vs shortest path routing

connections (n_{T_i}) that pass through itself. Furthermore, with a periodicity of T_H seconds, each node broadcasts n_{T_i} to its one-hop neighbors[9], in the form of HELLO messages. Thus, each node can compute its congestion weight using Equation (1). Note that this congestion weight is also piggybacked onto the HELLO messages. As in the CCAR scheme, in the DCAR scheme, 5 TCP connections are established between randomly chosen node pairs[10], in a static network. We set the weight of a TCP connection $W_{TCP} = 1$ and the link penalty $W_p = 1$. The periodicity of HELLO messages T_H is set to 1 second[11]. We modified the AODV protocol to make it congestion-aware. When a source node needs a route to a destination, it sends out an RREQ request. When an intermediate node receives the first copy of this RREQ, it sets an *RREQ collection timer* and collects all the copies of this RREQ that are received prior to the expiry of the timer. Upon the expiry of the timer, the intermediate node updates and forwards the RREQ that contains the lowest indicated congestion weight. Like the intermediate nodes, once the destination receives the first copy of the RREQ message, it sets an *RREQ collection timer* and collects all the copies of this RREQ that are received prior to the expiry of the timer. Upon the expiry of the timer, the destination responds with an RREP to the RREQ message that contains the lowest indicated congestion weight. Once the source receives the RREP, a path with the least congestion weight is established between the source and the destination.

[9]In order to avoid high overhead, the congestion weight is propagated only to one-hop neighbors instead of to all neighbors within the sensing range. If we were to propagate information with regards to a larger neighborhood size, the size of the HELLO packets increases significantly. We performed a few sample simulations that demonstrated that this further degraded the performance of TCP with the DCAR scheme.

[10]We also performed simulations with 10 TCP connections and the simulation results are similar to those with 5 TCP connections. We therefore omit these results from this paper.

[11]Simulations with T_H= 2 seconds and 3 seconds are also performed. As we will discuss later, choosing longer HELLO intervals causes stale information, which undermines the performance of the DCAR scheme. On the other hand, choosing shorter HELLO intervals incurs high overhead which has a significant negative effect on the performance of the DCAR scheme.

Figure 2 compares the performance, in terms of goodput, of the DCAR scheme with that of shortest path routing. We see that with the DCAR scheme, even though the goodputs of long TCP connections improve, the goodputs of short TCP connections suffer, on average, a decrease by about 5% to 10%. This observation is different from our observation in Section IV , in which the goodput of long TCP connections improves without hurting short TCP connections. The reason for the observed performance difference is that even though the DCAR scheme mimics the behavior of the CCAR approach, the DCAR scheme is still different from the CCAR strategy in the following ways:

- The DCAR scheme uses HELLO messages to allow nodes to exchange congestion weight information. In the CCAR scheme this information is provided to nodes magically, without overhead.
- In the CCAR scheme, the congestion weight is updated at each node instantaneously if there are any route changes (route breakages or route establishments) in the network. In the DCAR scheme, the congestion weights are updated by means of HELLO messages. The HELLO messages are broadcast only every T_H seconds. It is possible that the congestion weights of nodes stale if there are route changes that are yet to be identified or reported. Note that when a link breaks with AODV, the RERR message only propagates to the source. The nodes that are on the path between the broken link and the destination are oblivious to the route failure until these nodes receive another RREQ from the same source or their *connection alive timers*[12] for the connection expires. Furthermore, the changes in the congestion weights of nodes along a broken route cannot be traced by the neighbors of these nodes until the nodes report the changes in the HELLO messages of the next round.
- In the CCAR scheme, a *globally* optimal path in terms of the chosen congestion weights can be computed because we assume that each node knows the topology of the entire network and the paths used by on-going TCP connections. In the DCAR scheme, it is possible that the discovered path is not the globally optimal path. The reason for this is that the RREQ message that carries the optimal path information does not always reach the destination either due to RREQ collisions or due to the fact that the particular RREQ message is not received by intermediate nodes or the destination node before their RREQ collection timer (as discussed earlier) expires.
- In the CCAR scheme, the congestion weight of a node is computed to be the number of TCP connections passing through itself and its neighbors that are within its sensing

[12]The connection alive timer is used to trace the status of each TCP connection. The timer is updated once a node receives a packet that belongs to the associated TCP connection. If the timer expires, the node classifies the connection to be an expired connection and does not include it in the computation of its congestion weight. In the simulations, we set the value of the timer to be six times the average arrival interval (continuously estimated) of the packets that belong to the TCP connection.

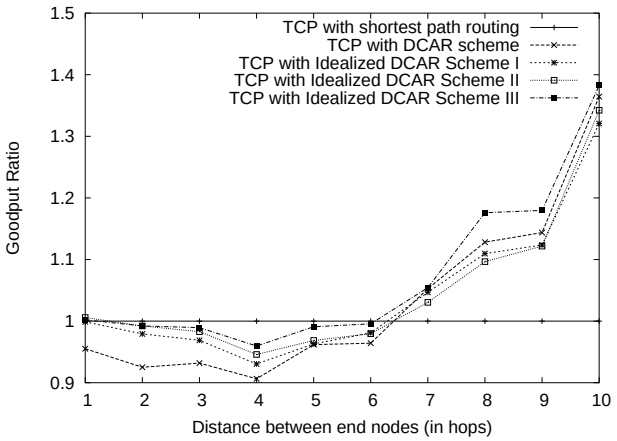

Fig. 3. Performance comparison of idealized versions of the DCAR scheme

range. In the DCAR scheme, in order to avoid high overhead (to control the size of the HELLO messages), the congestion weight of a node is broadcast to only its one-hop neighbors.

In the next sub-section, we show the effects of these differences on performance of the DCAR scheme by considering several idealized versions of the DCAR scheme.

A. Idealized Distributed Congestion-Aware Routing Schemes

In this sub-section, in order to evaluate the effects of the differences between the DCAR scheme and the CCAR scheme on TCP performance and to understand the reason for the DCAR scheme's poor performance, we consider several idealized versions of the DCAR scheme:

- *Ideal Scheme I*: In order to evaluate the effects of HELLO message on TCP performance, we consider an ideal case in which the congestion weight is exchanged among one-hop neighbors magically without overhead. We name it *magical hello* approach.
- *Ideal Scheme II*: In order to evaluate the effects of stale congestion information on TCP performance, we consider an ideal case in which the congestion weight can be updated instantaneously in a magical manner once there are route changes. We call this version, the *magical update* approach. In this ideal scheme, the congestion weight is also exchanged among one-hop neighbors magically without overhead (i.e., *magical-hello* approach is implicitly included).
- *Ideal Scheme III:* In order to evaluate the effects of the distance up to which the congestion weight is propagated, on TCP performance, we consider an ideal case in which the congestion weight can be magically distributed to the neighbors within the sensing range. In this ideal scheme, *magical hello* and *magical update* approaches are also included.

Figure 3 shows the TCP goodput ratio with these three idealized DCAR schemes (the comparison base line is still TCP with shortest path routing). We see that HELLO messages significantly hurt short TCP connections as compared

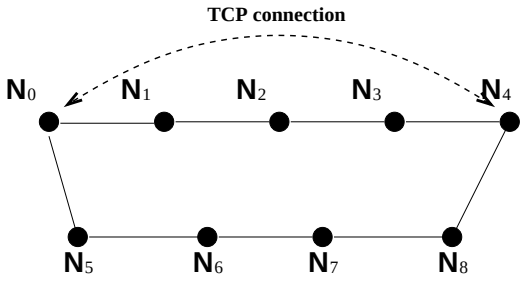

Fig. 4. A network with one TCP connection

to long TCP connections. In fact, the goodputs of long TCP connections improve with HELLO messages than without HELLO messages, thanks to the significant negative effects of HELLO messages on short TCP connections. Even though decreasing the frequency of HELLO messages may alleviate the aforementioned negative effects on TCP performance, longer HELLO interval causes stale congestion information that deteriorates the performance of TCP (we will discuss the effects of this factor later). Furthermore, by eliminating the effects of HELLO messages through the *magical hello* approach, we find that even though we improve the goodput of short TCP connections by a significant percentage, we still do worse than that with shortest path routing. In comparison, long TCP connections always obtain an increase in goodput.

In Ideal Scheme II, besides eliminating the effects of HELLO messages, we also eliminate the effects of stale congestion information by using the *magical update* approach. Stale congestion information is caused by the fact that, the congestion weights of the neighbors (of a tagged node under consideration) have been changed but since these neighbors are yet to broadcast information with regard to these changes, the tagged node under discussion is unaware of the changed weight. Here we use an example to illustrate how stale congestion information is caused and its effects on the route discovery process. There is one TCP connection from N_0 to N_4 in Figure 4. The solid line between nodes represents that they are one-hop neighbors. Suppose initially the connection uses path $N_0 - N_1 - N_2 - N_3 - N_4$. According to Equation (1), the weights of N_0, N_1, N_2, N_3 and N_4 are 2, 3, 3, 3 and 2, respectively. The weights of N_5, N_6, N_7 and N_8 are 1, 0, 0 and 1, respectively. If a false link failure occurs when N_0 tries to communicate with N_1, N_0 has to initiate a new route discovery process. In this case, even though N_0 knows that the path to N_4 has been broken, N_1, N_2, N_3 and N_4 still use the old congestion weights during the new route discovery process. Now, the weight of the path $N_0 - N_1 - N_2 - N_3 - N_4$ is larger than that of the path $N_0 - N_5 - N_6 - N_7 - N_8 - N_4$ and thus, the second path is chosen by the connection. Note that the second path is one hop longer than the first path. A longer path translates to lower TCP goodput (because there is only one TCP connection in this network). Note that with shortest path routing, the source could have possibly re-computed the original path via nodes N_1, N_2 and N_3. This effect is likely

to have a more dramatic impact on shorter TCP connections than the longer ones[13]. By eliminating the effects of this factor by means of the *magical update* approach, even though the goodput of short TCP connections can be improved, it is still less than that with shortest path routing. Since, the short TCP connections benefits from Ideal Scheme II, they end up injecting higher packet loads on to the network, This in turn, hurts the goodput (due to the increased congestion) of the longer TCP connections.

In Ideal Scheme III, besides eliminating the effects of HELLO messages and stale congestion information, we propagate the congestion weight of a node to all neighbors within its sensing range magically, without any overhead. We see that the goodputs of long TCP connections are improved and the short TCP connections are not affected as much. However, TCP with Ideal Scheme III still performs worse than TCP with the CCAR scheme. The main reason for the performance difference is that with the CCAR scheme, once a source node receives a route reply, it computes the *globally* least congested path and use it to communicate. With the DCAR scheme and the three idealized versions of the DCAR scheme, the "least congested path" being discovered may not be the *globally* least congested path. The RREQ message that carries the optimal path information may not reach the destination either due to RREQ collisions or due to the fact that the particular RREQ message is not received by intermediate nodes or the destination node before their RREQ collection timers (as discussed earlier) expire. The failure of discovering the globally least congested path causes the TCP connection to use a sub-optimal path to transmit packets. The use of such sub-optimal paths further prevents the DCAR scheme from obtaining the same benefits seen by the CCAR scheme.

B. Microscopic Performance Studies of Congestion-Aware Routing

In this sub-section we construct a microscopic example to provide a more fundamental understanding of the results discussed in the previous sub-section. In particular, our objective is to deliberate the reasons for the enhancement (in terms of goodput) in performance of the longer TCP connections and the complementary degradation in the performance of the shorter ones with congestion-aware routing.

We consider a grid topology shown in Figure 5. The distances between both the vertical neighbors and horizontal neighbors on the grid are 240 meters. There are two TCP connections in the network: TCP connection I is from N_{20} to N_{29} and TCP connection II is from N_{21} to N_{24}. These two TCP connections are initiated at the same time and they last for 120 seconds. The goodputs of the two TCP connections are

[13]As an example, if a single hop connection enjoys a capacity of C Mbps, by adding an additional hop immediately reduces to $C/2$ Mbps since only one of three nodes in the route can transmit at any given time. Similarly the addition of a fourth node (and a fifth node) can reduce the achievable maximum capacity to $C/3$ (or $C/4$) Mbps. Beyond four hops however, simultaneous transmissions of the same connection are possible and the reductions with additional hops are less dramatic.

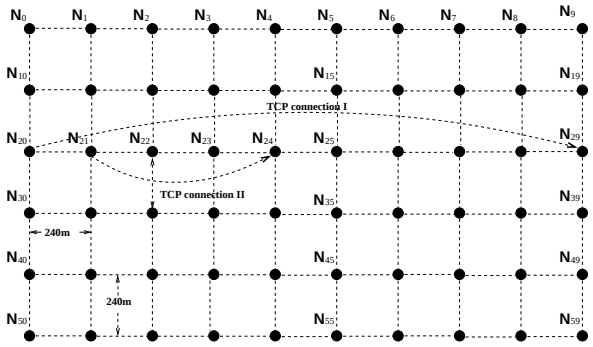

Fig. 5. A network with two strings and two TCP connections

computed with the shortest path routing scheme, the DCAR scheme and the idealized DCAR scheme III.

The simulation results tabulated in Table I are the average of 20 simulation runs with different random seeds[14]. From Table I we see that long TCP connection (Connection I) has the lowest goodput with shortest path routing while the short TCP connection (Connection II) has the highest goodput with shortest path routing. With the DCAR scheme, the long TCP connection obtains about a 74% improvement in its goodput while the short TCP connection suffers about a 14% decrease in its goodput. With Idealized DCAR scheme III, the goodput of the short TCP connection is almost the same as it is with shortest path routing. However, the goodput improvement obtained by the long TCP connection is lower with the Idealized DCAR scheme III as compared to that with the DCAR scheme. We further examine the routes that these two connections use with these schemes and find that with the shortest path routing scheme, most of the time, TCP connection I uses the route $N_{20} - N_{21}.....N_{24}.....N_{29}$, which is the shortest path for this connection. At the same time, TCP connection II always uses the route $N_{21} - N_{22} - N_{23} - N_{24}$, which is also the shortest path for it. With the DCAR scheme, TCP connection II still uses route $N_{21} - N_{22} - N_{23} - N_{24}$ while TCP connection I uses either route $N_{20} - N_{10} - N_0.....N_5.....N_9 - N_{19} - N_{29}$ or route $N_{20} - N_{30} - N_{40}.....N_{45}.....N_{49} - N_{39} - N_{29}$ in order to bypass the congestion region generated by TCP connection II[15]. In these two cases, N_1 through N_4 or N_{41} through N_{44} are within the congestion region generated by Connection II (for example, N_1 and N_{41} are within the sensing range of N_{21}). The interference caused by Connection II to Connection I is alleviated in this case than that with shortest path routing. Therefore, Connection I becomes more robust in the competition with Connection II. On the other hand, since Connection I becomes more robust, the interference caused by it to Connection II increases. The increased interference decreases the goodput of TCP Connection II. With the Idealized DCAR scheme III, we find that while TCP Connection II still

[14]Note here that even though the units of measurement are packets, each packet is of fixed length (1460 bytes) as mentioned earlier.

[15]The exact route chosen depends on the random back-off timers particular to the experiment.

TABLE I

GOODPUT OF THE TWO TCP CONNECTIONS WITH DIFFERENT ROUTING SCHEMES (UNIT: NUMBER OF PACKETS)

Routing Schemes	TCP Connection I	TCP Connection II
Shortest Path Routing	2882	226
DCAR	2516	395
Idealized DCAR Scheme III	2860	300

uses route $N_{21}-N_{22}-N_{23}-N_{24}$, TCP connection I uses route $N_{20}-N_{30}-N_{40}-N_{50}.....N_{55}.....N_{59}-N_{49}-N_{39}-N_{29}$. Note here that, with Ideal Scheme III, the weight of the nodes N_1 through N_4 and N_{41} through N_{44} are changed instantaneously (i.e., they reflect higher weights) with the initiation of the short TCP connection. Thus the longer connection chooses the correct least congested path, i.e., choose the path via node $N_{51}-N_{59}$. We wish to point out that with the DCAR scheme, since the node weights of N_1 through N_4 and N_{41} through N_{44} reflect a zero weight even with the short TCP connection (HELLO messages carry only one-hop congestion information), the long TCP connection is routed on one of these sub-optimal paths. In this case, the interference between these two connections occurs only at N_{20}, N_{21}, N_{22}, N_{30} and N_{40}. Therefore, we expect the long TCP connection to obtain some spatial separation benefits without significantly hurting the short TCP connection.

From this microscopic example we also observe that, with the congestion-aware routing scheme, short TCP connections have fewer routes to choose than long TCP connections. As an example, if Connection II in Figure 5 were to choose a path to circumvent (if possible) the longer connection and its interference zone, the weight on the new chosen path would significantly be higher than the weight of the shortest path. Thus, with each and every considered routing policy, Connection II is always routed on the same path which is the shortest path. In summary, it is difficult for short TCP connections to bypass congestion areas and obtain spatial separation benefits. Furthermore, long TCP connections become more robust and thereby compete better with short TCP connections. This causes the goodputs of long TCP connections to improve at the expense of hurting short TCP connections to a certain extent.

In a nutshell, even though we obtain performance gains by spatially separating TCP connections in a centralized ideal scenario, the additional overhead incurred for exchanging congestion information, stale congestion information, and lack of global information at each node prevent our DCAR scheme from exploiting such benefits. The performance results of our DCAR scheme are not specific in that, any other distributed congestion-aware routing scheme, in order to discover congestion-aware paths, will either (a) have to expend high overheads to obtain highly accurate congestion information, or (b) expend much lower overhead at the expense of obtaining much less accurate and possibly stale congestion information. In the first case, high overhead undermines their performance and in the second case, the reduced accuracy causes the use of sub-optimal path and thus prevents the exploitation of spatial separation benefits. Overall, any distributed congestion-aware routing scheme cannot completely eliminate the effects of the factors that were discussed earlier. In summary, we conclude that congestion-aware routing fails to provide significant spatial diversity benefits from TCP's perspective.

VI. CONCLUSIONS

In this paper, we show that the overhead that is necessary for achieving distributed congestion-aware routing in wireless ad hoc networks severely undermines the spatial diversity gains achieved. We perform a thorough investigation of whether spatial separation of TCP connections can provide benefits and explore a distributed way of achieving the gains. First, we investigate if we can obtain performance gains through separating TCP connections spatially in ad hoc networks, with a centralized ideal approach. Our studies suggest that the benefits due to spatial separation of TCP connections do exist. However, the benefits only apply to long TCP connections. Subsequently, we design a distributed congestion-aware routing scheme to exploit the benefits observed with the centralized ideal approach. The performance results with our distributed congestion-aware routing scheme show that even though the distributed congestion-aware scheme can help improve the goodput of long TCP connections, it is at the expense of hurting short TCP connections. We also consider several idealized versions of the distributed congestion-aware routing scheme to investigate (a) the effects of the overhead incurred for exchanging congestion information, (b) the effects of stale information, and (c) the effects of using sub-optimal paths (possible due to inaccurate information) on the performance of our distributed congestion-aware routing scheme. Our studies show that these factors prevent our distributed congestion-aware scheme from obtaining the benefits that were observed with the centralized ideal approach. The effects that were observed with our distributed congestion-aware routing scheme are not specific to our scheme and will affect any other distributed congestion-aware routing scheme. We conclude that achieving noteworthy performance gains by spatially separating TCP sessions may be extremely difficult if not impossible in ad hoc networks.

REFERENCES

[1] S. Lee and M. Gerla, "Dynamic Load-Aware Routing in Ad Hoc Networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 3206–10, 2001.

[2] H. Hassanein and A. Zhou, "Routing with Load Balancing in Wireless Ad Hoc Networks," in *Proceedings of the 4th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 89–96, 2001.

[3] K. Wu and J. Harms, "Load-Sensitive Routing for Mobile Ad Hoc Networks," in *Proceedings of the IEEE International Conference on Computer Communications and Networks (ICCCN)*, pp. 540–6, 2001.

[4] J. Song, V. Wong, and V. Leung, "Load-Aware On-Demand Routing (LAOR) Portocol for Mobile Ad hoc Networks," in *Proceedings of IEEE Vehicular Technology Conference*, 2003.

[5] C. Perkins and E. Royer, "Ad Hoc On-demand Distance Vector Routing," in *Proceeding of the Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA.*

[6] C. Cordeiro, S. Das, and D. Agrawal, "COPAS:Dynamic Contention-Balancing to Enhance the Performance of TCP over Multi-hop Wireless Networks," in *Proceedings of 10th Internation Conference on Computer Communication and Networks (IC3N)*, pp. 382–7, October 2002.

[7] V. Anantharaman and R. Sivakumar, "A Microscopic Analysis of TCP Performance over Wireless Ad-hoc Networks," in *Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems (poster paper)*, pp. 270–1, 2002.

[8] L. Wang, L. Zhang, Y. Shu, M. Dong, and O. Yang, "Adaptive Multipath Source Routing in Wireless Ad Hoc Networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2001.

[9] H. Lim, K. Xu, and M. Gerla, "TCP performance over Multipath Routing in Mobile Ad Hoc Networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 1064–8, 2003.

[10] K. Fall and K. Varadham, "The ns Manual." *http://www.isi.edu/nsnam/ns/ns-documentation.html/*.

[11] I. S. Department, ed., *Wireless LAN medium access control (MAC) and physical layer (PHY) specifications*. IEEE standard 802.11, 1997.

[12] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A Feedback-based Scheme for Improving TCP Performance in Ad Hoc Wilress Networks," *IEEE Personal Communications Magazine*, pp. 34–39, February 2001.

[13] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," in *Proceedings of the ACM International Conference on Mobile Computing and Networking (MobiCom)*, pp. 219–30, 1999.