Split TCP for Mobile Ad Hoc Networks

Swastik Kopparty, Srikanth V. Krishnamurthy, Michalis Faloutsos, Satish K. Tripathi
Department of Computer Science and Engineering, University of California, Riverside, Riverside, CA, 92521

***Abstract*—The fairness and throughput of TCP suffer when it is used in mobile ad hoc networks. This is a direct consequence of TCP wrongly attributing packet losses due to link failures (a consequence of mobility) to congestion. While this problem causes an overall degradation of throughput, it especially affects connections with a large number of hops, where link failures are more likely. Thus, short connections enjoy an unfair advantage over long connections. Furthermore, if the MAC protocol defined in the IEEE 802.11 standard is used, the problems exacerbate due to the capture effect induced by this protocol, leading to a larger degree of unfairness and a further degradation of throughput.**

In this paper we develop a scheme which we call *Split TCP*. This scheme separates the functionalities of TCP congestion control and reliable packet delivery. For any TCP connection, certain nodes along the route take up the role of being proxies for that connection. The proxies buffer packets upon receipt and administer rate control. The buffering enables dropped packets to be recovered from the most recent proxy. The rate control helps in controlling congestion on inter-proxy segments. Thus, by introducing proxies we emulate shorter TCP connections and can thereby achieve better parallelism in the network. As shown by our simulations, the use of proxies abates the problems described i.e., a) it improves the total throughput by as much as 30% in typical scenarios and b) it reduces unfairness significantly. In terms of an unfairness metric that we introduce, the unfairness decreases from 0.8 to 0.2 (1.0 being the maximum unfairness). We conclude that incorporating TCP proxies is beneficial in terms of improving TCP performance in ad hoc networks.

I. Introduction

Mobile ad hoc networks (MANETs) have received a lot of attention recently and may be deployed in military operations, disaster rescue missions and electronic classrooms. Thus, the need for interfacing MANETs with the Internet has arisen. TCP, which is the de facto transport layer protocol for the Internet, might be expected to be used for MANETs. TCP, however was designed for wire-line networks and significant degradations in its performance in terms of a significant reduction in the achievable throughput and an increased unfairness among connections are observed when it is used in MANETs. In the following paragraphs we elucidate these problems.

Several studies ([1], [2] and [3]) have shown that TCP cannot handle mobility well. In ad hoc networks, links can break causing temporary packet losses (until the routing layer discovers a new route). Furthermore, as the number of hops on a path increases, the probability of a link failure (and consequential packet losses) on the path increases. This implies that shorter TCP connections enjoy an unfair advantage in throughput as compared with longer connections[1]. In [1] and [2], it was proposed that explicit notifications be sent back to the source node when a link failure occurs on the path of that connection. The TCP source then freezes the connection until the route is restored. While this method can alleviate the problem to a certain extent, it does not overcome the unfair advantage that short sessions enjoy with respect to longer sessions. Furthermore, this scheme does not allow the utilization of the other links on the path that have not failed. Long connections are much more likely to freeze because (a) they have more links and are therefore are more susceptible to failure, and (b) since short connections can transmit faster (by adjusting their TCP window size more rapidly) they can dominate shared links.

The MAC protocol commonly used in ad hoc networks accentuates the performance problems of TCP [3]. This MAC protocol is the one described in the IEEE 802.11 standard [4] and we will refer to this as the 802.11 MAC protocol for short. One specific problem that is induced by the exponential backoff mechanism of the 802.11 MAC protocol, is the *channel capture effect*. Simply put, because of this effect, the most data-intense connection dominates the multiple-access wireless channel. If there are multiple data-intense connections, the first connection "captures" the channel until it has transported all of its data to the destination. This creates an unfairness to the connections that begin later or are further away from the point of contention. So as we see again, the longer connections are in disadvantage.

In this paper, we examine the effect of splitting long TCP connections into shorter *localized segments* to primarily improve the performance in terms of fairness. We substantiate this idea by developing a scheme that uses proxies as interfacing agents between these localized segments. We call this scheme *Split-TCP* or *TCP with proxies*. Intuitively, the best way to think of inter-proxy segments is as "zones"; once the packet makes it to a proxy it has traversed the previous "zone" and thus, we avoid having to go all the way back to the source if the packet needs to be retransmitted. More specifically, a proxy intercepts TCP packets, buffers them, acknowledges their receipt to the source (or previous proxy) by sending a local acknowledgement (LACK), and takes over the responsi-

This work was supported in part by grants from the Tata Consulting Services, Inc., the Digital Media Innovations Lab of the University of California and also by the DARPA award FTN F30602-01-2-0535

[1] We use the term "long" to refer to connections that are over long paths. They do not refer to connections of long durations.

bility of delivering the packets further, at an appropriate rate, to the next local segment. Upon the receipt of a LACK (from the next proxy or from the final destination), a proxy will purge the packet from its buffer. The forwarded packet could possibly be intercepted again by another proxy and so on[2]. In this scheme, we do not change the end-to-end acknowledgment system of TCP, meaning that the source will not clear a packet from its buffer unless it is acknowledged by a cumulative ACK from the destination. However, as we shall see later, the overhead incurred in including infrequent end-to-end ACKs in addition to the LACKs is extremely small, and can be considered to be accetable, given the advantages of Split TCP.

The main contribution of this work is that, by introducing Split-TCP, we split the transport layer functionalities into those of end-to-end reliability and congestion control. This is done in recognition of the fact that congestion tends to be a local phenomenon, specific to the environment, whereas reliability is an end-to-end requirement. We quantify the gain in performance due to Split-TCP through simulations.

In Section II, we describe the motivation for designing Split-TCP and discuss why it might be expected to improve throughput and alleviate the problems of unfairness among TCP sessions in an ad hoc network. In section III we present our simulation results and discuss their implications. Our conclusions form the final section.

II. An Overview of Split-TCP

In this section, we provide an overview of how TCP proxies work, and provide qualitative arguments that show the motivation behind their use.

Proxies split a TCP connection into multiple local segments. They buffer packets and deliver them to the next proxy or to the destination. Each proxy receives packets from either the source (A proxy P1 receives packets from S in Figure 1) or from the previous proxy, sends LACKs for each packet to the sender (source or proxy) of that packet (as an example in Figure 1, the second proxy P2, upon receiving a packet, sends a LACK for that packet to P1), buffers the packet, and when possible, forwards the packet towards the destination, at a rate proportional to the rate of arrival of LACKs from the next local segment. The source keeps transmitting according to the rate of arrival of LACKs from the next proxy, but purges a packet from its buffer only upon receipt of an end-to-end ACK for that packet (note that this might be indicated in a cumulative ACK for a plurality of packets) from the destination.

This essentially splits the transport layer functionalities into that of congestion control and end-to-end reliability. Correspondingly, we propose to split the transmission window at the source into two windows, the congestion window and the end-to-end window. The congestion window would always be a sub-window of the end-to-end window. While the congestion window changes in accordance with the rate of arrival of LACKs from the next proxy, the end-to-end window will change in accordance with the rate of arrival of the end-to-end ACKs from the destination. The dynamics of both these windows vary as per the rules that govern traditional TCP subject to the condition that the congestion window stays within the end-to-end window. At each proxy, there would be a congestion window which would govern the rate of sending between proxies. We suggest that these end-to-end ACK's be infrequent (one end-to-end ACK for every 100 or so packets that are received by the destination), since the likelihood of a proxy failure might be expected small[3].

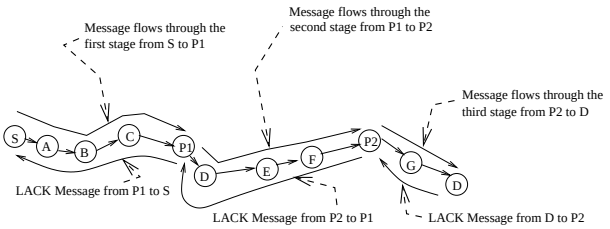

Fig. 1. TCP with proxies

We elaborate on the advantages of TCP proxies with regards alleviating the two effects that cause TCP to perform poorly: (a) mobility, and (b) the link capture effect of the 802.11 MAC protocol.

Dealing with Mobility: Split-TCP can handle mobility better than the plain TCP. Mobility in MANETs manifests itself as link failures. As the length (in hops) of a particular session increases, the possibility of link failures on that path also increases. One link failure can cause an entire TCP session to choke, when in fact packets can be transferred on other links that are still *up*. Split TCP helps take advantage of these links that are *up*. When a link on a local segment fails, it is possible for TCP with proxies to sustain data transfer on other local segments. Thus, the hit on TCP throughput due to mobility is of much lower impact.

We point out that the higher probability of link failures on longer paths (as mentioned) causes an unfair disadvantage to long TCP sessions when compared with shorter TCP sessions. By splitting the long TCP session into shorter local segments[4], we essentially create a scenario in which all TCP sessions are of short length. Thus, we can expect that our scheme improves the fairness among TCP sessions in the network.

Dealing with the link capture effect: If the IEEE 802.11 MAC protocol is used in conjunction with TCP, it causes the *channel capture effect*. If we have two simultaneous TCP sessions that are initiated in the geographical vicinity of each other, and are both heavily loaded, this effect provides an un-

[2]Note that TCP proxies (or similar entities) have been used successfully in cellular wireless networks [5], [6]. A major difference in our work is that the dynamic placement and maintenance of proxies is a challenging issue; in cellular networks the proxies can be conveniently placed at the base station. Furthermore, our proxies do not assume any responsibility of end to end reliable delivery.

[3]The only time the receipt of a LACK by the source does not mean that the packet is delivered end-to-end is when either a proxy fails or the network becomes disconnected.

[4]Local segments can be thought of as short TCP connections, where the ACKs correspond to LACKs. Henceforth, we shall freely use the analogy in our discussion.

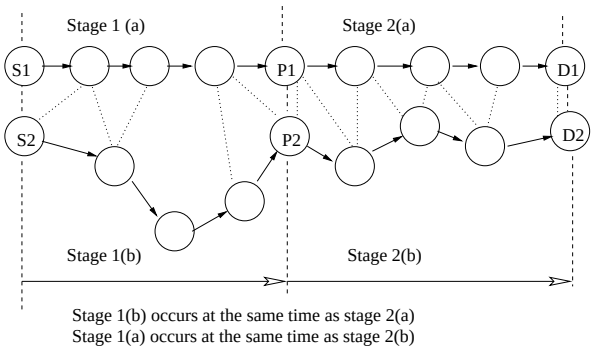

Fig. 2. The method by which TCP with proxies solves the channel capture effect

fair advantage to the session that originated earlier or to the session that is of fewer hops. The session with the advantage captures the entire region of transmission as its TCP window size grows. The introduction of TCP proxies as described, alleviates this problem. Since, the streaming of data now occurs in shorter *stages*, only one stage is captured at a time i.e., the size of the captured region is now reduced. This may be explained with the help of Figure 2. We show two heavily loaded connections and the first session begins earlier than the second[5]. In the absence of proxies the first connection enjoys a sustained streaming of data at the MAC layer. As long as the source S1, continues to have data, it *captures* the region of transmission and thus, the second connection can get virtually no throughput. If, for the first session, we introduce a proxy P1 between the source S1 and the destination D1 as shown in the figure, the TCP session is now broken up into two segments; the first terminates at P1 and the second originates at P1. This division causes streaming to occur in two stages. During the first stage data is streamed from S1 to P1 and during the second stage data is streamed from P1 to D1. These two events would be mutually exclusive since, when node P1 is streaming data to D1, it will be unable to receive packets from S1, and when P1 is receiving packets from S1, it cannot stream data to D1. Thus, a second *Split TCP* session (like one from S2 to D2) can simultaneously stream packets in the geographical vicinity of the segment from S1 to P1, (that is from S2 to P2), as P1 is in the process of delivering its packets to D1. Similarly, as S1 streams packets to P1, P2 can complete its delivery of packets to D2. Thus, by splitting TCP what we essentially ensure is that only small geographical areas in the network are captured for MAC access and for short times, and therefore, we improve fairness among the TCP sessions. The advantage is more pronounced for long TCP connections that contain multiple proxies.

We envision that proxies can be implemented with no per-flow state. Every node would maintain a buffer that can store the aggregated packets from every flow for which the node is a proxy. It also maintains a *local congestion window*, that is a reflection the amount of space left in this aggregate buffer. The source would then regulate its rate of flow based upon the minimum of the local congestion levels at the proxies on its path. We omit the details of this implementation due to the lack of space and refer the reader to [7].

Proxies would be implemented by means of a distributed algorithm. Each node would look at the number of hops that a packet has traversed since the previous proxy (as seen in the IP header) and if a predetermined hop count is reached, it acts as a proxy for that packet. Such an implementation would make deployment of nodes with proxy capabilities both practical and scalable [6].

III. Experimental Results

In this section we quantify the performance enhancements that Split-TCP provides in ad hoc networks by means of extensive simulations. We use the popular ns 2.1 and the extensions provided therein for supporting mobile ad hoc networks [8]. We have created a "TCP proxy agent" and a "TCP connection manager" module to support the operation for Split-TCP.

To compare Split-TCP with simple TCP, we use the cumulative throughput which is defined as the ratio of the total number of data bytes transported to the destination over the duration of the session as one of the metrics. We use a second metric for quantifying the unfairness, U, which stems from a commonly used fairness metric [9]. Let us assume that there are k simultaneous connections at a given time (let these connections be enumerated from 1 to k). Let us also suppose that the cumulative throughput, at this given time, for a connection i is t_i. We also define T_i to be the cumulative throughput that connection i would have enjoyed if it had been the *only* connection that was active. We define the unfairness to be

$$U = \sqrt{\frac{k\sum_{i=1}^{k}(\frac{t_i}{T_i})^2 - (\sum_{i=1}^{k}\frac{t_i}{T_i})^2}{k-1}}. \quad (1)$$

If all the $\frac{t_i}{T_i}$ are between 0 and 1, then the value of U is also between 0 and 1. Note that $U = 1$ is the maximum unfairness, while $U = 0$ is fair. The reason for taking the ratio of the throughputs during simultaneous and isolated operations is to measure the unfairness that occurs due to the presence of other simultaneous connections[7]. For example, if $k = 2$, then $U = |\frac{t_1}{T_1} - \frac{t_2}{T_2}|$.

A proxy node upon receiving a data packet, *first* forwards the data packet to the next relay node and then generates a LACK message to be sent to the previous proxy (or source)[8]. In our experiments, unless mentioned otherwise, we consider that the source of a TCP session is data-intensive, i.e., it always has packets to send. We conducted several experiments varying the inter-proxy distance and found that an inter-proxy

[5] A similar reasoning may be made if one of the connections is heavily loaded as compared with the other.

[6] Due to space constraints we cannot provide further details on possible implementation methods; these are provided in [7]

[7] We use this mainly to demonstrate that our method alleviates the unfairness that arises due to the capture effect of the IEEE 802.11 MAC protocol.

[8] Reversing the order of these two events resulted in reduced performance.

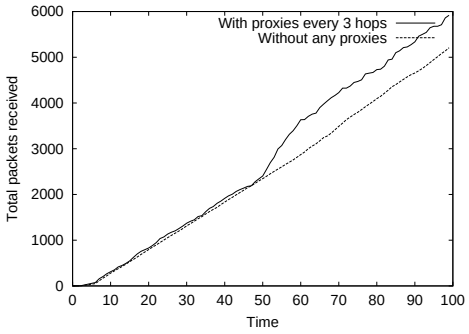

Fig. 3. Variation of total packets received of a typical connection with time in a mobile ad hoc network

distance of 3 or more is good for throughput and and inter-proxy distance of approximately 5 or less is good for the fairness. Hence, in the following experiments, we use an inter-proxy distance of 3, 4 or 5 unless explicitly stated otherwise.

We consider a mobile network of 50 nodes spread uniformly in a 1km x 1km square region. Each node can move with a speed which is chosen according to a uniform distribution and varies from 0 to 10 m/s (the random waypoint model is used). We arbitrarily choose source-destination pairs and TCP sessions are set up between them. We run every scenario for both TCP and Split-TCP. We run several experiments with 3, 4 or 5 simultaneous TCP connections. The overall observation is that **the total throughput improves with the use of proxies by about 5 % to about 30%**. Furthermore, proxies serve as moderators in the sense that faster connections are slowed down and slower connections are made faster, i.e., fairness is improved.

First, we examine the dynamics of how proxies improve the performance of TCP when there is mobility. We present the results of a typical experiment with 3 TCP connections in Figure 3 where we plot the total throughput in number of packets received versus time. Observe that the throughput with proxies increases in a burst in the small duration of time between 50 and 60 seconds. The reason for this behaviour was that, at this time, one of the connections (which we will call connection 1) which was of 5 hops, had a link failure (due to mobility) at the second hop. The proxy was positioned after the third hop. The connection had no route to the destination for a short period of time. TCP without proxies experienced losses and kept reducing its window size by half after each packet loss was seen. However, in the case of Split-TCP, though the source-proxy TCP segment was throttled, the proxy- destination TCP segment continued to function. It is interesting to note that not only was the instantaneous throughput maintained, but it in fact increased, since now, the proxy- destination segment did not have to compete for channel access, in the vicinity of the proxy, with the source - proxy segment. Since there was a considerable number of packets queued at the proxy, the connection with proxies continued to function. Note that the total throughput of TCP without proxies does not change. This is because, although one of the connections is cut off, the other two connections hogged the channel with their own data.

During the time that the connection 1 encountered a failed link, the throughput achieved by this connection by using TCP with no proxies, is zero. The two other connections, on the other hand, enjoy an enhanced throughput at its expense. By using our unfairness metric defined in Equation 1, we see that this causes an average unfairness of 0.24. On the other hand, when Split-TCP was used, Connection 1 continued to get a fair share of the throughput in spite of the link failure. The value of the unfairness metric was now reduced to 0.13. Although we have shown results from one of our experiments, this improvement in throughput and fairness was typical for all simulation runs. These simulations were run with the number of nodes varying from 50 to 100, and the maximum speed of each node varying from 5 m/s to 20 m/s. **T**he average (over all simulations) increase in throughput was about 12%. The average (over all simulations) value of the unfairness metric observed for TCP (without proxies) was 0.47, while the average (over all simulations) value of the metric for TCP with proxies (inter-proxy distance = 3 or 4 hops) was 0.17.

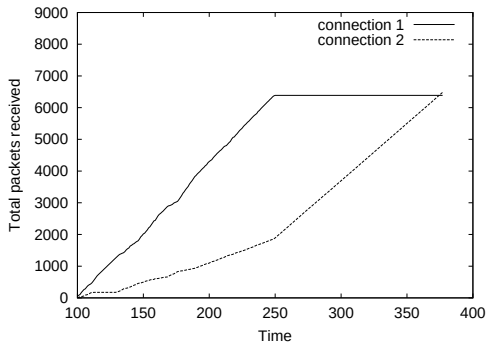

Fig. 4. Individual Throughputs of Competing TCP sessions: Regular TCP

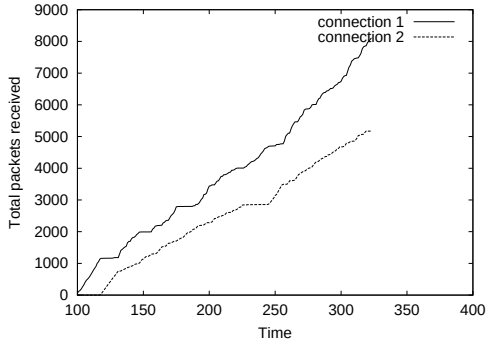

Fig. 5. Individual throughputs of competing TCP sessions: Split TCP

Split-TCP alleviates the Unfairness of an Early Start: When we have two or more sessions that are active simultaneously, and have along their route, a region wherein the channel is shared between them, we find that with TCP, the value of the unfairness metric is large. Consider Figure 4;

for this experiment we had two TCP connections, connection 1 and connection 2. Connection 2 began slightly later than connection 1. We see that connection 2 is almost at a standstill until connection 1 completed sending all its data. On the other hand, when Split TCP was used, as shown in Figure 5, the throughput of connection 2 was about the same as that of connection 1.

We recall our discussion of the effect of the IEEE 802.11 MAC protocol on TCP from section II. Since connection 1 began earlier, it captured the channel to begin with. Connection 2, continually experienced exponential back-off at the MAC layer. As explained in section II (Figure 2), Split-TCP helped alleviate this effect and improved the share of the channel access that was available to connection 2.

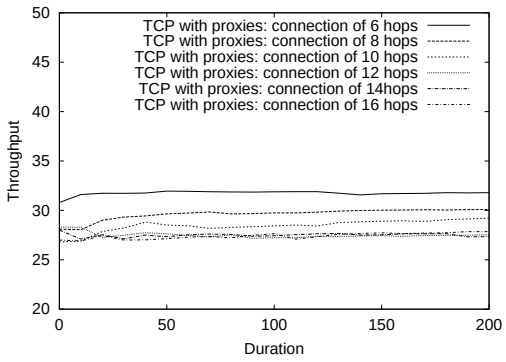

Fig. 6. Effects of duration (secs) and number of hops on throughput (KB/s): TCP with proxies every three hops

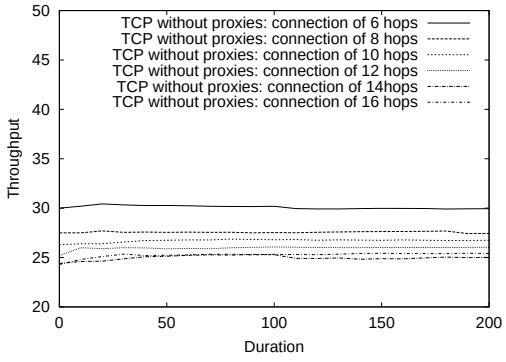

Fig. 7. Effect of Duration (sec) and Number of hops on Throughput (KB/s): TCP Without Proxies

Next, in Figures 6 and 7, we plot the throughput of a connection as a function of the connection duration and its length for both TCP and Split-TCP. As expected, longer connections benefit by using Split-TCP (for a connection that is 14 hops in length, Split-TCP provides a throughput gain of about 16%). The connection duration seems to have little effect on the performance, regardless of whether TCP or Split-TCP is used. Notice that a TCP connection of shorter length always outperforms a TCP connection that is longer. This is due to the fact that there is always some contention among adjacent links (whether proxies are used or not) for channel access time. Thus, the larger the number of hops, the higher the contention, and hence, lower the throughput.

Split-TCP is robust to proxy failures. We have also performed experiments in which proxies failed or became disconnected from the network. In such cases, the end-to-end ACKs helped in end-to-end packet recovery. It was observed that even if proxies fail, the overall throughput is higher than that of regular TCP (provided that the rate of proxy failure is not too high i.e., occur only about 5 % of the time) [9], while at the same time, the end-to-end reliability of TCP is preserved.

IV. Conclusions

In this paper, we propose a new promising approach to improve the performance of TCP in terms of fairness and throughput in MANETs. We propose to achieve this by introducing proxy agents that split TCP into localized segments. Our new version of TCP is called Split TCP. The proxy agents facilitate the separation of the congestion control and the end-to-end reliability semantics of TCP. When a link failure occurs in one segment, sustained throughput is possible on the other segments. Since, link failures are more probable in longer connections than in shorter ones, the introduction of proxy agents especially benefits longer connections. To summarize, TCP proxies *succeed* in terms of achieving a higher TCP throughput and providing better fairness to longer TCP connections with respect to shorter ones. In addition, the introduction of proxies help alleviate the *channel capture* effect that occurs if the IEEE 802.11 MAC protocol is used. If plain TCP is used, the channel capture occurs over the entire region occupied by a heavily loaded TCP session whereas with Split-TCP the effects are now localized. We show by means of simulations that Split TCP can improve both the fairness among TCP connections (by a factor of 60%) and the throughput (by about 5% to 40%) of individual TCP connections.

References

[1] G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobile Ad Hoc Networks," in *IEEE/ACM MOBICOM*, August 1999.
[2] K.Chandran *et al*, "A Feedback based Scheme for Improving of TCP Performance in Ad hoc Wireless Networks," in *IEEE Personal Communications Magazine*, February 2001.
[3] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC Protocol Work Well in Multihop Wireless ad hoc networks," in *IEEE Communications*, June 2001.
[4] "Draft International Standard ISO/IEC 8802-11, IEEE P8.2.11/D10," in *LAN MAN Standards Committee of the IEEE Computer Society*, January 1999.
[5] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, 1997.
[6] A. Bakre and B.Badrinath, "I-TCP, Indirect TCP for Mobile Hosts," in *15th International Conference on Distributed Computing Systems (ICDCS)*, 1995.
[7] S.Kopparty *et al.*, *Split-TCP for Ad Hoc Networks*, UC Riverside Technical Report., 2002.
[8] ns 2.0 Network Simulator, "http://www.isi.edu/nsnam/ns," .
[9] Raj Jain, *The art of computer systems analysis*, John Wiley and Sons, 1991.
[10] Y.Ko and N. Vaidya, "Using Location Information in Wireless Ad Hoc Networks," in *IEEE Vehicular Technology Conference*, 1999.

[9] Due to lack of space we do not present those results here.