Available online at www.sciencedirect.com

SCIENCE DIRECT®

Ad Hoc Networks 4 (2006) 60–74

Ad Hoc Networks

www.elsevier.com/locate/adhoc

Using a new protocol to enhance path reliability and realize load balancing in mobile ad hoc networks [☆]

Antonios Argyriou [*], Vijay Madisetti

School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-1070, USA

Received 5 February 2004; accepted 14 April 2004
Available online 11 May 2004

Abstract

In this paper we introduce a novel end-to-end approach for achieving the dual goal of enhanced reliability under path failures, and multi-path load balancing in mobile ad hoc networks (MANETs). These goals are achieved by fully exploiting the presence of multiple paths in mobile ad hoc networks in order to jointly attack the problems of frequent route failures and load balancing. More specifically, we built a disjoint-path identification mechanism for maintaining multiple routes between two endpoints on top of the Stream Control Transmission Protocol (SCTP), and the Dynamic Source Routing (DSR) protocol. A number of additional modifications are incorporated to the SCTP protocol in order to allow its smooth operation. The proposed approach differs from previously related work since it consists of an entirely end-to-end scheme built on top of a transport layer protocol. We provide both analytical and simulation results that prove the efficiency of our approach over a wide range of mobility scenarios.
© 2004 Published by Elsevier B.V.

Keywords: Mobile ad hoc networks; SCTP; DSR; Reliability; Load balancing

1. Introduction

The advent of mobile ad hoc networks provided an efficient, and most important cost effective way of exploiting the presence of mobile hosts when no infrastructure is available. Numerous are the challenges that this class of wireless networks set to the research community [1]. A considerable number of these challenges have been met through the development of sophisticated routing protocols which through their simplicity can provide stable solutions in these environments [2–4]. However, little research has been done at the higher protocol layers in order to take advantage of this highly versatile environment for further improving the delivered end-to-end performance. Research

[☆] A preliminary and short version of this paper appeared in WCNC 2004.

[*] Corresponding author. Tel.: +1 404 483 8328.
E-mail addresses: anargyr@ece.gatech.edu (A. Argyriou), vkm@ece.gatech.edu (V. Madisetti).

1570-8705/$ - see front matter © 2004 Published by Elsevier B.V.
doi:10.1016/j.adhoc.2004.04.005

efforts focused at the upper layers have primarily focused on piecemeal modifications at specific transport or application layer protocols (e.g. TCP) so that the protocol behavior matches as close as possible to the one of wireline networks [5,6]. However, there is a lack of systems that introduce solutions that offer a new perspective to the problems related to mobile ad hoc networks. This claim does not necessarily mean the design and deployment of completely new protocols since this is not a realistic option. However, there are classes of optimizations that can be done when we realize the full potential of these networks. Some of these optimizations do not arise in conventional centralized wireless access systems. The specific feature that is of interest to us, is the existence of multiple paths between two endpoints in an ad hoc network. Of course, multipath routing has been studied extensively (see Section 2). A number of proposals exist, that make use of multiple paths for either improving route reliability or for load balancing. Route reliability by itself is very crucial in ad hoc networks since their main feature are the frequent link breakages due to the mobility of every node. Current research efforts have focused on masking these effects to the upper layer protocols. On the other hand, load balancing mechanisms for MANETs, are based on modifications of well known routing protocols such as DSR [2–4].

However, in this paper we propose a novel idea in order to approach the above problems: we claim that maintaining multiple paths and using them at the transport layer, the overall system performance can be improved without any modifications at the routing layer. We show that by maintaining multiple active paths at the transport layer, a mobile host can recover faster in the case of route failures. We also show that when the transport layer takes over the load balancing process, it can deliver better end-to-end performance since it is capable to mask effectively the various side-effects of load balancing over multiple paths. We incorporate our ideas as a number of algorithms and modifications to the SCTP protocol [7]. More specifically, we devise a new algorithm for disjoint path identification, and present a number of modifications to the algorithms used by the SCTP protocol. Even if it might seem that our system addresses two different problems, our claim is that we rather exploit the full potential that the existence of multiple paths gives in an MANET in a unified framework.

The rest of this paper is organized as follows: Section 2 provides an overview of the related work. Section 3 gives an overview of SCTP and DSR since these are the protocols that we are using throughout this paper. In Section 4 we analyze our method for locating disjoint paths and using them for improving path reliability. Section 5, provides an analysis of the load balancing system. In Section 6 we considered useful to provide an analytical evaluation of our path monitoring mechanism. In Sections 7 and 8 we provide simulation results for both the proposed subsystems. Finally Section 9, concludes the paper.

2. Related work

Current routing protocols for MANETs such as DSR and TORA, are capable of providing multiple paths between the source and the destination. However, no processing is performed in order to identify if these paths are disjoint. The idea of identifying disjoint paths for multipath routing has been studied by a number of researchers [8–13].

One of the first and most comprehensive works has been reported by Nasipuri et al. [8]. The authors develop analytical formulas in order to model the effect of the number of multiple paths and lengths of those paths on the performance of DSR. Another interesting piece of work towards this direction was presented by Leung et al. [10]. The authors of this work optimize DSR by defining a new metric that is used as a measure of the end-to-end path reliability. Purpose of their main algorithm is to identify disjoint paths between two endpoints, so that in the case of a path failure there will be alternatives for usage. In addition, they allow the user to specify the required reliability level, disallowing thus a number of paths to be discarded. However, they do not address load balancing for application layer data. Recently Ye et al. [13], suggested a modification to AODV that uses multiple paths between the source and the destination. However, the proposed system relies heavily on the existence of specialized and highly-reliable nodes in the ad hoc network, requirement which is impractical for civil applications. Another recent work presented by Xue et al. [14], proposes a theoretically derived approach to the problem of fault tolerant routing. They prove that the problem is NP-complete and they present an estimation-based fault tolerant routing algorithm. Nevertheless their system results into high overhead, since it duplicates the same data packet over the number of multiple paths that are used. Another approach reported in [15], performs a theoretical evaluation of the problem of load balancing at the routing layer, without providing any implementation. The main conclusion is that that the data distribution across the various routes must be performed according to the RTT values of each route.

Concerning other approaches at higher protocol layers, there is no significant piece of work. However, we find advisable to mention that there is some recent works that present performance evaluation of the SCTP protocol over mobile ad hoc networks [16,17].

3. Overview of SCTP and DSR

In this section we provide an overview of the protocols that we are using in this paper: DSR and SCTP. A thorough assessment of the DSR protocol can be found in [18], while a detailed experimental evaluation can be found in [19,20]. The SCTP RFC [7], is currently the best reference for this new protocol.

3.1. DSR overview

DSR is a simple source routing protocol for MANETs, in which route caching is heavily used. If the route to the destination is not known, a route discovery process is initiated in order to find a valid route. Route discovery is based in flooding the network with route request (RREQ) packets. Every mobile host that receives a RREQ packet checks the contents of its route cache, and if it is the destination or it has a route to the destination it replies to the RREQ with a route reply (RREP) packet that is routed back to the original source. In case none of the above holds, the host that received the RREQ re-broadcasts it to its neighborhood. In this way the RREQ message is propagated till the destination. Note that both RREQ and RREP packets are also source routed. The RREQ message maintains the path traversed across the network allowing thus the RREP message to route itself back to the source by traversing the recorded path backwards. The route carried back by the RREP packet is cached at the source for future use. If any link on a source route is broken, the source host is notified with a special route error (RERR) packet from intermediate nodes. When the source gets this packet removes any route using this link from its cache. More details and enhancement to this basic DSR operation can be found in [2,18].

3.2. SCTP overview

SCTP was recently adopted by IETF, and is a reliable transport protocol that operates on top of a connectionless packet based network such as IP. One of the most important new ideas that SCTP introduced is that of multi-homing. A single SCTP association (session), is able to use alternatively anyone of the available IP-addresses without disrupting an ongoing session. However, this feature is currently used by SCTP only as a backup mechanism that helps recovering from link failures. SCTP maintains the status of each remote IP address by sending Heartbeat messages and it is thus able to detect a specific link failure and switch to another IP address. Another novel feature is that SCTP decouples reliable delivery from message ordering by introducing the idea of *streams*. The *stream* is an abstraction that allows applications to preserve in order delivery within a *stream* but unordered delivery across *streams*. This feature avoids HOL blocking at the receiver in case multiple independent data streams exist in the same SCTP session. Congestion control was defined similar to TCP, primarily for achieving TCP friendliness [7].

4. Improving path failure recovery

4.1. Disjoint path identification

Essential condition for our system to be effective, is the existence of multiple paths between the two mobile hosts that want to communicate. Currently, DSR implementations maintain a number of routes for a specific destination in the route cache. This is true because the DSR route discovery process generates a number of route replies (RREP packets).[1] In this way, the system is capable of selecting paths for failure recovery or load balancing, from a number of alternatives. However, not all of them are equally good for use. This is because we set the important requirement that the paths which will be used must be node-disjoint and not just edge-disjoint. This requirement might sound very strict but as we will see in the next paragraph, it is necessary for the smooth operation of the end-to-end protocol.

Consider for example the case of load balancing. In this case the goal is to simply distribute the traffic across the available routes so that the mobile host uses all the available aggregate bandwidth. However, if the paths are edge but not node-disjoint it is possible that sub-flows that belong to the same end-to-end aggregate session, will merge at the same host which may end up being a bottleneck. Packets may be dropped and the following interesting sideffect occurs: packets from different sub-flows, but of the same main flow, will end up antagonizing with each other leading to reduction of the congestion window in each path which means under-utilization of the available bandwidth. That is why we require node-disjoint paths to be existent in the caches. There are recent studies that have studied extensively this sideffect in the case of the Internet [21]. In that case, the authors have found that the probability of two sub-flows sharing a common bottleneck router is pretty low [21]. However, this is not the case now since ad hoc networks are obviously characterized by small scale deployment. In the next section, we present our algorithm that is capable of finding these node-disjoint paths.

4.1.1. The algorithm

Fig. 1 provides a simplified version of the algorithm that is responsible for finding disjoint paths between two hosts. The algorithm operates as follows: When there is a request from the application to send a message/packet to *Destination i*, this algorithm checks the cache that contains routes that are disjoint and valid for load balancing (*LCache*). If there are no routes there, it checks the DSR routing cache (*RCache*), and adds the results to the *LCache*. However, there is a case when no route exists in both of the caches and so a route discovery process must be initiated. Now, when a valid route has been found, the next step is to check if the paths that correspond to the destination *dest*, have any common nodes (i.e. they are not node-disjoint). If there is such a case, paths with high RTT are removed from the *LCache*. In this way we only keep in the *LCache* only the paths with the lowest RTT for future reference. This pruning process goes down until we have a number of non-overlapping routes equal to the number of desired maximum number of paths that will be used for load balancing. In this way this heuristic provides a fast way for obtaining the best available paths. Even if it is rather obvious that we always prefer routes with low RTT, there are additional

```
if $[paths = FIND(LCache, dest)] \neq \emptyset$
    continue
elseif $[paths = FIND(RCache, dest)] \neq \emptyset$
    then $ADD(LCache, Destination\ dest, paths)$
    else $SEND(RREQ)$
endif

for each Path $(i, j) \in LCache \rightarrow dest$
if $LCache \rightarrow dest \rightarrow i \bigcup LCache \rightarrow dest \rightarrow j \neq \emptyset$
        $tmp = MAX(LCache \rightarrow dest \rightarrow i \rightarrow rtt,$
                $LCache \rightarrow dest \rightarrow j \rightarrow rtt)$
    $REM(LCache, tmp)$
else continue
endfor
```

Fig. 1. Disjoint path selection algorithm. After the algorithm is finished the data-stripping process is initiated.

[1] Even though there is a mechanism for preventing RREP implosion [2].

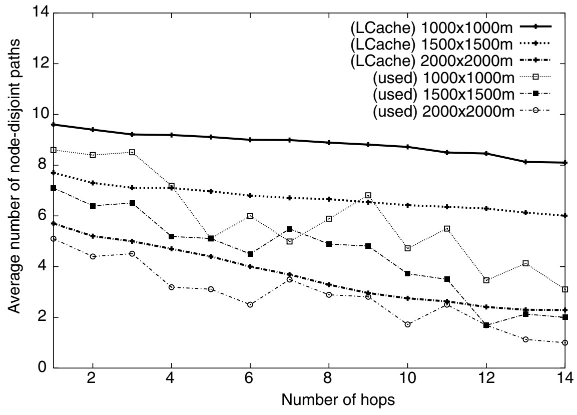


Fig. 2. Disjoint paths for 400 mobile hosts.

reasons which will be explained later as we delve more into the transport layer issues that affect our system. Note, however, that routes are not removed from the route cache (*RCache*) even when they have high RTT, since this value may momentarily change.

Fig. 2 depicts clearly this fact through some interesting results. We see that the average number of node-disjoint paths, that are available in the *LCache*, is relatively small. Previous research also validates that [13]. As we use larger and larger area, the number of available disjoint routes is decreasing. We also show with additional curves which paths from the ones reside in the *LCache*, were actually used. These paths were obtained after a simple test was performed, Eq. (8), in order to avoid large RTT mismatches. Basically we will later see the fact that whether this inequality holds, determines the actual number of paths used for load balancing. Now, because these routes depend on the value of RTT of each disjoint path, they are different when movement scenarios change. That is why we observe a large fluctuation in these curves, and a smaller degree of dependence from the number of hops.

4.2. Protocol modifications for enhancing robustness in case of route failures

Simply identifying node-disjoint paths is not enough for SCTP to either behave better in case of route failures or implement load balancing. A number of modifications to the base protocol are necessary for proper operation. For this part of the system, primary purpose is to improve the robustness of the connection and mask the effects of frequent link breakages to SCTP. These modifications are necessary for the smooth operation of a transport layer protocol in an ad hoc network environment [5]. Throughout the next sections, we analyze these problems and propose solutions to them. So, in order to achieve the above, we identified the need for modifications in two SCTP algorithms: The path heartbeating and packet retransmission algorithms (only during changeover).[2]

4.2.1. Modifying the heartbeating mechanism

According to the SCTP RFC [7], each endpoint of an SCTP association can configure a number of addresses. This can be true for both the source and the destination. A number of four handshake steps are involved before data transfer is initiated by SCTP between the primary local and primary remote addresses. After that, and during the lifetime of the association, SCTP sends Heartbeat messages that monitor the status of all the secondary remote addresses. Note that SCTP cannot monitor the actual path status but only the destination address, since it does not have network layer information concerning the exact routing path.

In order to resolve the above problem, we modified the SCTP protocol so that the sender before it sends a packet, it ensures that packets destined for the same mobile host (but for different address) are sent to different paths. The modified SCTP assigns the best path, which was obtained after execution of the algorithm in Fig. 1, to the main data flow, while for the time being, the only packets sent to the secondary paths/addresses are Heartbeat messages (Fig. 3). By default, SCTP will start sending Heartbeat messages periodically (every RTO) to every destination address. Since the heartbeating mechanism involves a Heartbeat ACK packet, we can have explicit feedback about the status of a specific path. The crucial difference of a vanilla

[2] According to SCTP terminology, changeover refers to the re-assignment of the outgoing traffic to another interface [22].

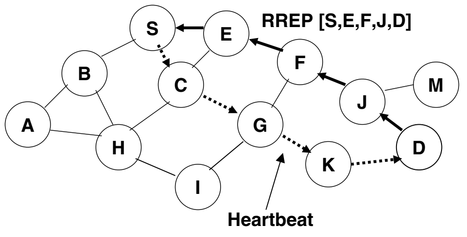


Fig. 3. Heartbeating mechanism.

DSR/SCTP version and ours, is that DSR would inform with a RERR message only if the primary path was down. The happens to be also true for popular mechanisms like TCP with explicit link failure notification [5]. However, in our DSR/SCTP optimized system, we maintain a small number of redundant paths for which we get network statistics every RTO. The trick here is that we "force" some of the paths to be alive by sending Heartbeat messages. The only cost, is the slight probing overhead due to the heartbeating mechanism, but as we will later see is negligible.

Another issue that has to do with heartbeating, is the decision concerning the time for changeover. In order to do that, SCTP uses the Heartbeat timer, which is maintained for Heartbeat messages sent to every destination address. Five retransmissions is the value used by vanilla SCTP for deciding that this time has come [7]. Nevertheless, we found that we could be more elastic with that and reduce this value to just two retransmissions (after 2RTO time). Why is that? Note that we already have an alternative route by DSR for SCTP and so switching pro-actively to this one would not do any harm. However, when the SCTP association has only one route available, SCTP reports failure to the upper layer after the usual five retransmissions. Moreover, we observed that while mobility is increased (in terms of pause time), timer values higher than 2RTO indicate route breakage and not packet loss. The reason for that is that it is unlikely that two consecutive (every RTO) Heartbeat messages were lost due to wireless errors.[3] Another reason for using 2RTO as the threshold for changeover, is because of the SACK mechanism used by SCTP for loss identification and recovery. Because of the SACK algorithm, SCTP is able to recover without RTO expiration when packet losses from a window of data are less or equal to half the congestion window [23].

[3] We tested packet loss rates 0.1–1%.

4.2.2. Avoiding unnecessary route discovery

The DSR protocol indicates that when a current route fails, the sender should initiate the route discovery process in order to find a new route to the destination. Measurements show that this process may last more than the RTO timer at the sender [6], resulting in a series of actions that degrade performance. Assuming that the above happens, the result will be an RTO timeout, packet retransmission and reduction of the congestion window. Even after a new route has been found, a TCP or SCTP sender will need some time in order to catch up with its previously achieved throughput, since it must increase the congestion window using slow-start. Moreover, in highly dynamic networks with highly mobile hosts and frequent link breakages, the above behavior will occur frequently resulting in poor performance.

So the enhancement in this case resides in the fact that we practically move "one layer up" the actual route selection, that is performed at the transport by the modified SCTP. By maintaining multiple routes provided by DSR, the transport layer has one more degree of flexibility concerning the actual route for its packet. One advantage of this approach is that we avoid the lengthy route discovery process in case of route failures [24], unless of course all the routes fail simultaneously. Otherwise, even if there is just a single alternative route, the SCTP sender immediately starts using it. Even if we follow the approach reported in [5], by using explicit link failure notification, the transport protocol's freezed state will not correspond after a while to the new path status but to the old one. All the state parameters will be invalid. However, in SCTP we maintain congestion control parameters for each path or source/destination address pair, allowing thus the maintenance of a distinct state for each path. Furthermore, our system is able to keep these parameters up to date, because of the heartbeating mechanism described earlier.

4.2.3. Overcoming changeover side-effects: packet reordering

We saw in the previous section that a DSR-based sender usually initiates the route discovery process and after it receives a valid route, it sends the packets to the destination. A problem that might show up here is the out-of-order delivery of the packets at the receiver, due to the asymmetry of the paths. This will trigger duplicate ACK (SACK for SCTP) transmission from the part of the receiver which will inevitably result in an invocation of congestion control at the sender. This is a major problem with multipath routing protocols such as TORA [6].

The problem of packet reordering can be appropriately tackled by careful optimization in the context of SCTP and DSR. The modified protocol is capable of doing this because it has exact knowledge of the time the sender switches to an alternative path provided by DSR. The modified SCTP basically captures RERR messages send to DSR that only concern failure of the primary route so that it can switch to an alternative path. Our version of SCTP, would also do that by itself as we earlier said after 2RTO timer expires on the primary path. By using this dual route failure detection mechanism, we are sure that the time of sender idleness will be minimum. This is clearly a transient behavior which SCTP enters when changeover is initiated. Fig. 4 presents the modified SCTP state machine so that it overcomes this transient packet reordering.

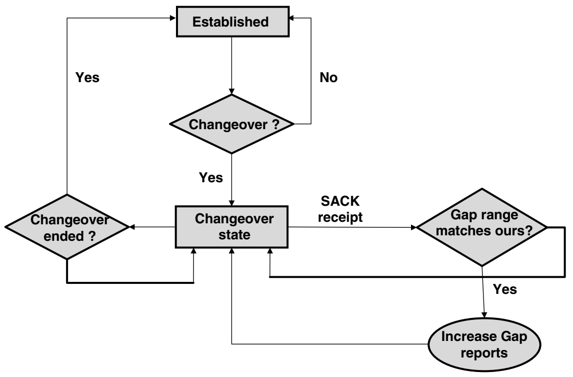


Fig. 4. Modified SCTP state machine.

In our implementation, the SCTP sender maintains two variables that keep the lowest and highest TSN sent during the last *cwnd* to the receiver. When the receiver replies with an SACK containing a Gap report for TSNs that do not belong to this range (see Fig. 4) the sender does not increase the Gap Ack reports and process the SACK chunk as a normal SACK that acknowledges the outstanding data for this transport address. The rationale behind this is that the chunks that the receiver is requesting were not send by its address were sent by the new address. In the meantime the receiver monitors its *CumTSNack*[4] and when data are received from an address fill the gap then the receiver stops sending gap reports with SACK messages.

5. The case of load balancing

Before we move on, we have to clarify that we do not consider any MAC layer related problems in this paper. We assume the usage of the base IEEE 802.11b as the MAC layer protocol from now on. However, there is an important issue that motivates our load balancing solution. According to work related to IEEE 802.11 modeling [25,26], the maximum system throughput that can be achieved by the DCF function, does not depend on the number of nodes that content for the medium. The maximum system throughput is given by [25]

$$S_{\max} = \frac{E[P]}{T_s + \sigma K + T_c(K(e^{1/K} - 1) - 1)}, \tag{1}$$

where $E[P]$, T_s, T_c, and σ are constants and K is another constant that depends on whether we use the basic access mechanism or the RTS/CTS enhancement. Assuming also that the DCF function achieves fairness in the longterm for all the participating hosts [27], we can see that the share of the bandwidth that a mobile host will get is linearly decreasing while the number of mobile hosts

[4] Cumulative TSN Ack Point: equivalent to TCP's largest sequence number received.

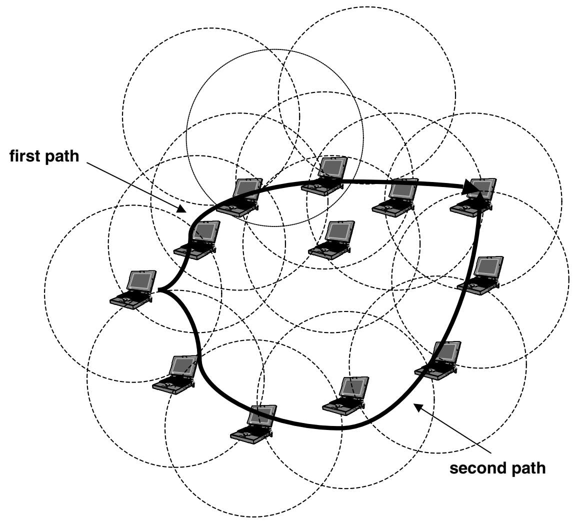


Fig. 5. Different paths with different contention levels.

increase. Now, let us have a look at Fig. 5 where we show an ad hoc network topology at a particular moment in time. We have indicated two valid paths that DSR has provided to the source host. We can see that the shortest path, which consists of four hops, passes through a dense area of hosts, while the path with five hops simply consists of a chain of nodes. The default DSR operation would be to use the first path [2]. However, the dense area suffers from high contention, and thus it restricts the available bandwidth for the source host, to a small portion of the total bandwidth. But the second path, which will have lower RTT, can perform better due to the low medium contention. Nevertheless there is a subtle point here: the second path is more amenable to a route failure since it consists of a simple chain of hosts which can easily break with the movement of a single host. On the other hand, the first path can still operate (and with better throughput) if a host moves out of the dense area.

The analysis that we just described, provides a very important conclusion and a design guideline for our system: It is possible due to the high loaded medium in the vicinity of the intermediate hosts, that a single end-to-end connection achieves a maximum throughput that is determined by this bottleneck. Despite that, higher density of mobile hosts is translated to better reliability under path failures. That is why, in our system, we do not throw away any of these node-disjoint paths but use them at the same time in order to implement load balancing, since they will be useful in either way.

5.1. Protocol modifications for efficient load balancing

The necessary modifications that implement load balancing add up to the ones mentioned earlier, without affecting already existing functionality. So, after the primary goal is achieved, that of identifying disjoint paths fast, we must make sure that we mask the effects of concurrent data transmission and frequent link breakages from the application layer. More specifically we made two modifications to the SCTP data sending process.

5.1.1. Efficient data stripping across multiple links

We mentioned earlier a number of load balancing techniques for ad hoc networks which all unfortunately share a common drawback. The problem is that in case of connection oriented protocols such as TCP, packet losses in one path (with low RTT) will throttle back the aggregate pipeline despite the fact that load balancing is taking place at the routing layer [21].

One crucial difference of our approach compared to related work, is that a single data flow is distributed to each outgoing path, according to the congestion window value of each specific transport address. We do not use the bandwidth of each link in order to estimate the amount of data that should be assigned. The rationale behind this is to fill the bandwidth-delay product of each link. In this way, we can also solve the problems that occur when the used links are highly asymmetric in terms of delay or available bandwidth [21]. In [28], the authors are following a similar approach but after they logically separate the sender in a number separate TCP sender entities in order to be able to apply congestion control for each link. In our case we handle all this in a unified way at the base SCTP protocol. One additional difference is that we maintain data in flight for all destinations. However, data are assigned to an interface after we apply congestion control for a particular destination. In this way data can be sent immediately. By following this approach, we avoid the need for dynamic data reassignment in case we have stale data for a particular destination which has decreased its congestion window.

Now, assume that N is the number of paths used by the mobile host. If we assume that each outgoing packet has the same size, the fraction of data that must be sent in path i in order to avoid spurious retransmissions will have to be [29]

$$f = \frac{cwnd_i}{cwnd_{\mathrm{tot}}}, \tag{2}$$

where $cwnd_i$ is the congestion window for path i and $cwnd_{\mathrm{tot}}$ represents the total data in flight from all the used paths. If a transport layer connection sends more data than this equation specifies in the corresponding link, then data will not delivered soon enough, and the limit of three duplicate acknowledgements at the sender will soon be reached and the sender will fast retransmit segments that are still in transit [7]. We follow this simple rule for stripping data to outgoing interfaces from now on. Note that before the data-stripping process starts, the algorithm of Fig. 1 is executed. This is achieved easily because we have implemented the aforementioned algorithm as a hook at the transport layer.

5.1.2. Avoiding RTO timeouts

Despite the fact that the algorithm of Fig. 1 will be able to find disjoint paths in case they exist in the DSR route cache, this does not automatically qualify them for using them for load balancing. The problem that arises is the following: Assume that two paths are found between the source and the destination and are disjoint. Assume also that their RTT differ by a factor of five (i.e. they are highly asymmetric). If packet with id 1 is transmitted to the slow link the rest four packets with id 2, 3, 4, and 5 must be sent to the faster link. Upon reception of the packets 2, 3, and 4 the receiver will send duplicate acknowledgments to the sender, which will trigger the retransmission process [7]. In this way the first transmission is useless and even worse the sender throttles back its congestion window [29]. This effectively means that the whole aggregate connection will be limited by the slowest link present.

An important observation is that since ad hoc networks have local scale and the hosts are very close to each other, the RTT between two nodes is dominated by the transmission delay, while all the other components (signal propagation, processing delays, ACK delays) are negligible [1]. Now we will provide an analytical evaluation of the necessary conditions for avoiding this problem. SCTP uses the same formulas for calculating the RTO timer value for each outgoing packet. So for packet i, RTO is

$$\mathrm{RTO}_i = \mathrm{SRTT}_i + 4 * \mathrm{RTTVAR}_i, \tag{3}$$

while SRTT and RTTVAR are

$$\mathrm{SRTT}_i = (1 - \mathrm{RTO}\cdot\alpha) * \mathrm{SRTT} + \mathrm{RTO}\cdot\alpha * R', \tag{4}$$

$$\mathrm{RTTVAR}_i = (1 - \mathrm{RTO}\cdot\beta) * \mathrm{RTTVAR}_{i-1} + \mathrm{RTO}\cdot\beta * |\mathrm{SRTT} - R'|\,. \tag{5}$$

Because we consider all the packets with the same size, the fraction of packets will also be the same $f = n_i/n_{\mathrm{total}}$. So the average values for both RTT and RTT variation ($\mathrm{RTT_{DEV}}$) when considering all the links are

$$\overline{\mathrm{RTT}} = \sum_{i=1}^{N} \overline{\mathrm{RTT}}_i \times \frac{n_i}{n_{\mathrm{total}}}, \tag{6}$$

$$\overline{\mathrm{RTT_{DEV}}} = \sum_{i=1}^{N} (\mathrm{RTT}_i - \overline{\mathrm{RTT}}_i) \times \frac{n_i}{n_{\mathrm{total}}}. \tag{7}$$

Basically every time RTT value changes (it is updated every RTT for each path) for anyone of the links used in load balancing, the average $\overline{\mathrm{RTT}}$ is recalculated. Now, in order to avoid RTO timeouts, the RTT for every link must be

$$\mathrm{RTT}_i < \overline{\mathrm{RTT}} + 4 * \overline{\mathrm{RTT_{DEV}}}. \tag{8}$$

If this constrain is not satisfied then the specific route is not used for load balancing even if it is in the cache (*LCache*). On the other hand, if the constraint is satisfied, the route is immediately used. When there is a need to send a new packet, the algorithm in Fig. 1 will compare RTT between all the paths for a given destination in order to make sure again that Eq. (8) is satisfied. Note that we do not remove a route from the *LCache* even if it does not satisfy Eq. (8), since RTT might soon change and make the route usable.

6. Analytical modelling of the heartbeating mechanism

We think that it is necessary first to evaluate theoretically the heartbeating mechanism, since it can generally contribute significantly to performance degradation.

Section 3.3.5 of the SCTP RFC [7], describes the format of a Heartbeat request message. The chunk type, chunk flags, and chunk length constitute four bytes. Additional information that is piggybacked to the chunk, is in the form of TLV (Variable-Length) and it included sender specific related information. This information is basically the sender's current time when the Heartbeat message is sent and the destination address to which the message is sent. We assume that the size of of the time-stamp is 4 bytes and the destination address IP/Port are 4 and 2 bytes, respectively. So the total number of bytes is: 22 bytes or 5+1/2 "units". Moreover, according to the SCTP RFC [7], to an idle destination address that is active, a Heartbeat message is recommended to be sent once per RTO of that destination address plus a user configurable protocol parameter HB.interval. An exponential back-off algorithm of the RTO is applied when a previous Heartbeat message remains unacknowledged. So assuming that a path has an average value of RTO and the user parameter is HB.interval, the sender will send Heartbeats to a path with frequency 1/(RTO +HB.interval). Assuming that the number of destinations is N, and the average number of hops between the source and the destination is $L/2$ then the total unit cost will be (a unit = 4 bytes)

$$N\left(\frac{1}{\mathrm{RTO}+\mathrm{HB.interval}}\right)\left(\frac{L}{2}+6\right). \tag{9}$$

Heartbeat messages are regular DSR source routed data packets. A Heartbeat/DSR packet will have a size (in units)

$$\frac{L}{2}+6. \tag{10}$$

The corresponding overhead of an MAC layer packet is 82 bytes [30]. So the above becomes in bytes per second

$$N\left(\frac{1}{\mathrm{RTO}+\mathrm{HB.interval}}\right)\left(4\left(\frac{L}{2}+6\right)+82\right) \tag{11}$$

with a total number of packets/s

$$\frac{1}{\mathrm{RTO}+\mathrm{HB.interval}} \tag{12}$$

We get the static and dynamic DSR overheads from [30] and add the heart-beating related component so that we obtain the total signalling cost. For a static network in packets/second the DSR overhead is [30]:

$$f_{\mathrm{D}}\lambda N\left\{1+(1-f_{\mathrm{D}}^{\Delta})f_{\mathrm{D}}\Delta+f_{\mathrm{D}}^{\Delta}\left[N+\frac{\rho_{\mathrm{D}}L}{2}\right]\right\}, \tag{13}$$

where f_{D} is the probability that a route does not exist in DSR's cache, Δ is the average number of neighbors of a host, and ρ_{D} is the number of RREP packets for an outgoing DSR RREQ. While mobility induced losses are [30]

$$\frac{\mu_{\mathrm{D}}\alpha NL}{2}, \tag{14}$$

where μ_{D} is the link breakage rate due to mobility, and α is the number of active routes per host.

Fig. 6 depicts analytical results concerning the DSR and DSR/Heartbeating related overheads

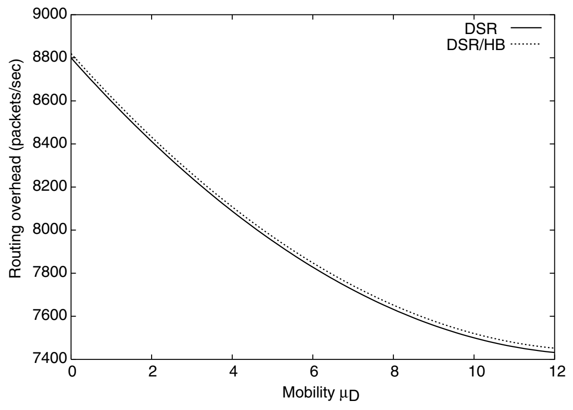


Fig. 6. DSR heartbeating overhead after analytical evaluation.

with respect to host mobility (combined Eqs. (12)–(14)). We can see that indeed, heartbeating does not incur too much overhead since the number of packets sent are relatively small, and also their size is small. We will see from simulation results, that indeed the slight increase in this signalling overhead offers a significant increase in the overall performance of end-to-end connections.

7. Evaluation of the modified SCTP/DSR with route failure recovery

In this section we provide simulation results after we implemented all the previously mentioned algorithms and modifications. The NS-2 network simulator [31], was used for all our experiments. We used 802.11 as the MAC level protocol and selected as the radio model Lucent's WaveLAN with range of 250 m [4]. DSR and SCTP are already integral part of the NS-2 simulator.

We defined a 600 m $\times$ 600 m area for host movements. The simulation time is 200 s and we used various pause times: 0, 10, 20, 40, 60, 80, 100, 150, and 200, that represent different levels of mobility. We used the widely accepted random way-point model for creating all the host movement scenarios [18]. In a movement scenario, each host starts by being initially stationary for a duration of *pause* time (s). After this time frame, the model requires it to select a random destination in the 600 m $\times$ 600 m space and start moving to that destination at a speed between 0 and some maximum speed. When it reaches the destination, the host pauses again for *pause* time seconds, selects another destination, and proceeds there as previously described, doing this behavior for the whole duration of the simulation. We experimented with three different maximum host speeds (10, 20 and 30 m/s). The majority of the results, unless otherwise specified, corresponds to host speed of 20 m/s. Backlogged FTP sources were used throughout all our scenarios. We kept these parameters fixed, since we believe that they represent a very typical testbed setup, and we tried a large number of alternate mobility scenarios.

7.1. Simulation results

Fig. 7 presents the progress of sequence numbers in case of TCP-Reno and vanilla-SCTP. Fig. 8 presents sequence number progression at the receiver for the same scenario but for the modified SCTP. We can see now that the modified SCTP/DSR does not suffer from the timeout of Fig. 7(b). And the reason is that at time 60 s, after 2RTO the modified SCTP sender switches to an existing alternative path. This also happened for other flows in the same scenario.

The average throughput for the modified protocols is shown in Fig. 9(a). Note that these results were obtained for the same 600 m $\times$ 600 m scenario with 50 hosts. It is clear that indeed the proactive nature of our modified algorithms it is able to take

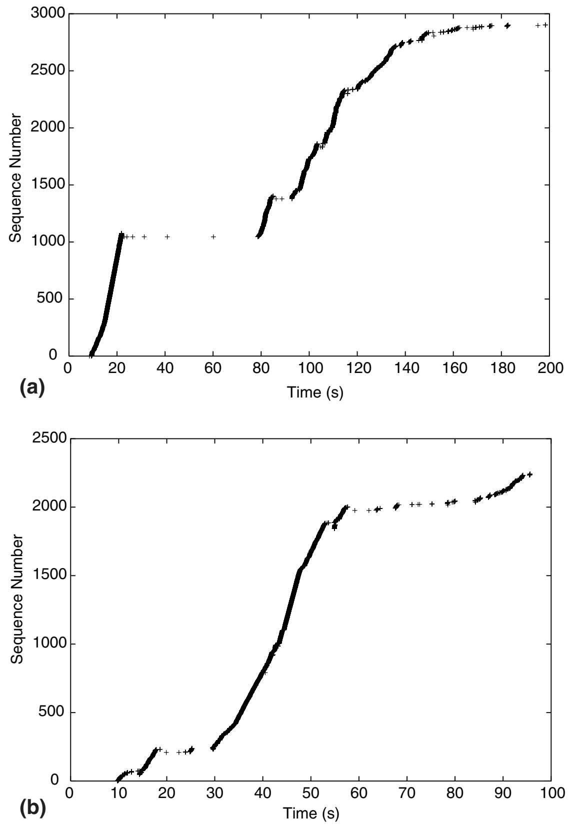


Fig. 7. Sequence number progression for the flow 18–20: (a) TCP-Reno and (b) SCTP.

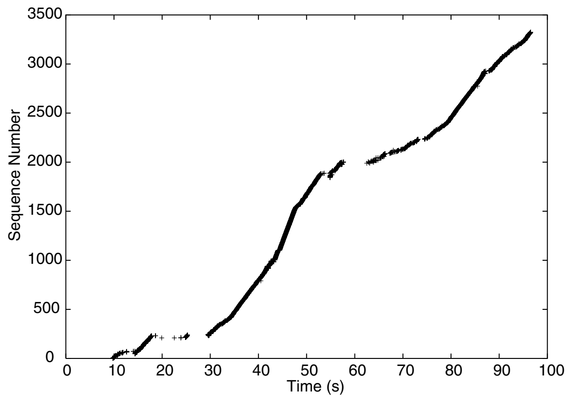


Fig. 8. Sequence number progression for the flow 18–20 with modified SCTP.

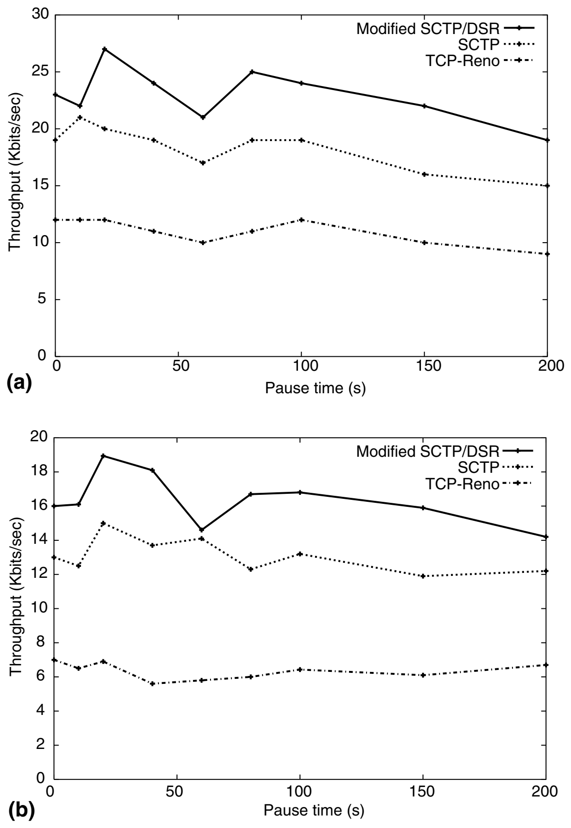


Fig. 9. Modified SCTP, SCTP and TCP throughput with underlying DSR for varying pause time: (a) 600 m × 600 m area and (b) 800 m × 800 m area.

advantage of alternative paths and switch to them as soon as possible.

However, in Fig. 9(b) we show results for a scenario with the same number of hosts but for area of 1000 m × 1000 m. Purpose of this experiment is to see the behavior of our modified protocol when the number of existing routes is smaller due to the large movement area. We can indeed see that performance of all the protocols is significantly reduced. The number of route failures is increased and the number of alternative paths becomes smaller. That is why both standard DSR and the modified SCTP over DSR perform worst.

8. Simulation results for SCTP with load balancing

The simulation setup that we used in this set of experiments is the same as before. In our first experiment we limited the maximum application data rate per connection to 33 Kbps, in order to see effects of our system at low bit-rates. Fig. 10 shows results for one specific data flow between two hosts. Three plots are shown: the ideal average aggregate throughput, aggregate throughput with modified SCTP/DSR, and throughput for unmodified SCTP/DSR (one link). It is clear that our system achieves good aggregate throughput. Even though during the simulation the movement pattern was kept the same, the number of concurrent links with neighbor hosts is very low for high mobility scenarios while it is quite increased for

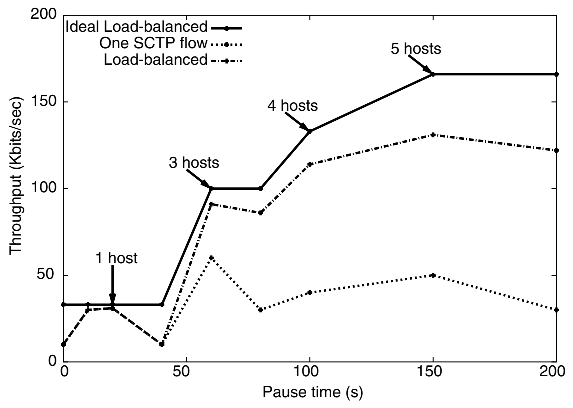
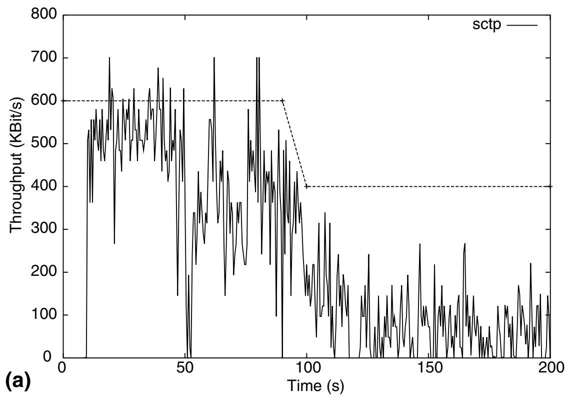


Fig. 10. Aggregate throughput for the modified SCTP. The mobile host is using five routes.

low mobility cases. The arrows in Fig. 10 indicate the average number of available disjoint paths with neighbor hosts that can be used for concurrent transmission. For example for high mobility scenarios (pause time between 0 and 40 s), the average number of hosts in the neighborhood that can be used in only one. It is obvious that the ideal throughput is directly dependent upon this parameter. However, this aggregate throughput is not possible to be achieved for one flow since there are other flows that content for the medium in this topology.

Fig. 11(a) shows the instantaneous aggregate throughput for one flow of one simulation round of 200 s. In addition we show, the available aggregate bandwidth according to the number of available links as before. For a significant amount of time the host captures nearly the full amount of the available bandwidth (three hosts × 200 Kbps). After time 100 s, the host has moved in a less dense area of hosts in which the two hosts that they exist there they have some incoming data flow. This results in the use of low part of the bandwidth (around 1/3). Note that this time the aggregate available bandwidth is three times lower than previously even though only one host has left the neighborhood.



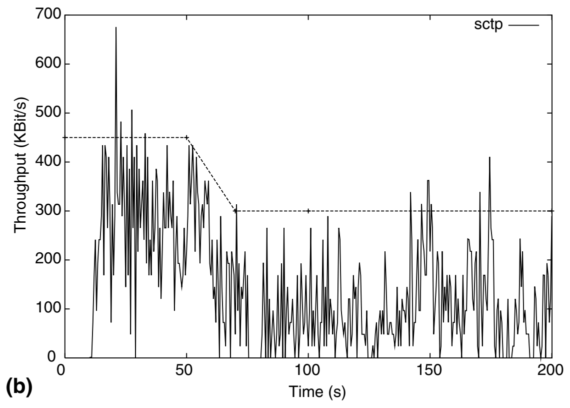


Fig. 11. Instantaneous throughput for two different flows of the modified SCTP system: (a) local contention after 100 s prevents the host from using the full aggregate bandwidth and (b) modified SCTP takes advantage of the low loaded medium for load balancing.

Fig. 11(b) shows the opposite situation where the host moves to a less dense topology again (from three to two hosts in neighborhood). The ideal rate is around 150 Kbps per link in this experiment. Even though the case seems like Fig. 11(a), the hosts that reside in the new place do not have any active flows which means that the load balancing system is able to take advantage of the additional paths with no contention. Nevertheless, before time instant 70 s, the host enjoyed better throughput since one more inactive host was present.

So the important conclusion is the following: more "dense" topologies result in the existence of more routes in the route cache allowing greater flexibility in the route selection process. However, when at the same time the number of flows are increased from these hosts, contention is increased and so the throughput of the aggregate connection will perform according this contention limit.

9. Conclusions

In this paper, we presented a system for better path reliability and load balancing in ad hoc networks. The proposed algorithm and modifications were implemented as part of the SCTP protocol. Our system assumes the use of DSR as the routing protocol. The basis of our system are a lightweight path monitoring mechanism for handling route failures pro-actively and a set of simple modifications to SCTP. By switching path pro-actively at the transport layer, we showed that we can achieve better performance by avoiding RTO timer expirations. The overall overhead of maintaining redundant paths is shown to be relatively small when compared to the overall throughput increase. In addition, we showed that load balancing is possible to be achieved in these diverse networks even if we do not have access to multiple wireless interfaces. Overall, the number of necessary modifications to the protocol is small, making thus our solution a good candidate for practical applications in ad hoc networks.

References

[1] I. Stojmenovic, Handbook of Wireless Networks and Mobile Computing, Wiley, New York, 2002.

[2] D.B. Johnson, D.A. Maltz, Y.-C. Hu, The dynamic source routing protocol for mobile ad hoc networks (DSR). Available from: <http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-09.txt>.

[3] E.M. Royer, C.K. Toh, A review of current routing protocols for ad hoc mobile wireless networks, IEEE Personal Communications 6 (2) (1999) 46–55.

[4] C.E. Perkins, E.M. Belding-Royer, S.R. Das, Ad hoc on-demand distance vector (AODV) routing. Available from: <http://www.ietf.org/internet-drafts/draft-ietf-manet-aodv-13.txt>.

[5] G. Holland, N. Vaidya, Analysis of TCP performance over mobile ad hoc networks, in: Proceedings of IEEE/ACM Mobicom 1999.

[6] J. Liu, S. Singh, ATCP: TCP for mobile ad hoc networks, IEEE Journal of Selected Areas in Communications 19 (7) (2001) 1300–1315.

[7] R.R. Stewart et al., Stream control transmission protocol, RFC 2960, October 2000.

[8] A. Nasipuri, R. Castaneda, S.R. Das, Performance of multipath routing for on-demand protocols in mobile ad hoc networks, Mobile Networks and Applications 6 (4) (2001) 339–349.

[9] K. Wu, J. Harms, Performance study of a multipath routing method for wireless mobile ad hoc networks, in: Proceedings IEEE International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2001, pp. 99–107.

[10] R. Leung, J. Liu, E. Poon, C. Chan, B.Li, MP-DSR: A QoS-aware multi-path dynamic source routing protocol for wireless ad hoc networks. Available from: <http://citeseer.nj.nec.com/557864.html>.

[11] S.J. Lee, M. Gerla, Split multipath routing with maximally disjoint paths in ad hoc networks, in: Proceedings IEEE ICC 2001, pp. 3201–3205.

[12] M. Pearlman et al., On the impact of alternate routing for load balancing in mobile ad hoc networks, in: Proceedings of the ACM Mobihoc, August 2000.

[13] Z. Ye, S.V. Krishnamurthy, S.K. Tripathi, A routing framework for providing robustness to node failures in mobile ad hoc networks, Ad Hoc Networks 2 (2004) 87–107.

[14] Y. Xue, K. Nahrstedt, Fault tolerant routing in mobile ad hoc networks, in: Proceedings of WCNC 2003, pp. 1174–1179.

[15] L. Zhang et al., Load balancing of multipath source routing in ad hoc networks, in: Proceedings of the IEEE ICC 2002.

[16] G. Ye, T. Saadawi, M. Lee, SCTP congestion control performance in wireless multi-hop networks, in: Proceedings of MILCOM 2002, Anaheim, CA, October 2002.

[17] A. Argyriou, V. Madisetti, Performance evaluation and optimization of SCTP over mobile ad hoc networks, in: Proceedings of the 28th Local Networks Conference (LCN), Bonn, Germany, October 2003.

[18] D.B. Johnson, D.A. Maltz, Dynamic source routing in ad hoc wireless networks, in: T. Imielinski, H. Korth (Eds.), Mobile Computing, Kluwer Academic Publishers, Dordrecht, 1996, pp. 153–181 (Chapter 5).

[19] C. Perkins, E. Royer, S. Das, M. Marina, Performance comparison of two on-demand routing protocols for ad hoc networks, IEEE Personal Communications 8 (1) (2001) 16–28.

[20] J. Broch, D.A. Maltz, D.B. Johnson, Y.C. Hu, J. Jetcheva,A performance comparison of multi-hop wireless ad hoc network routing protocols, in: Proceedings of the ACM/IEEE MOBICOM 1998.

[21] A. Argyriou, V. Madisetti, Bandwidth aggregation with SCTP, in: Proceedings of the 2003 Global Communications Conference (GLOBECOM), San Franscisco, CA, December 2003.

[22] J.R. Iyengar et al., SCTP congestion window overgrowth during changeover, in: SCI 2002, July 2002.

[23] K. Fall, S. Floyd, Simulation-based comparisons of tahoe reno and SACK TCP, ACM Computer Communication Review 26 (3) (1996).

[24] J. Monks, P. Sinha, V. Bharghavan, Limitations of TCP-ELFN for ad hoc networks, in: Proceedings of MOMUC, Tokyo, Japan, October 2000.

[25] F. Cali, M. Conti, E. Gregori, IEEE 802.11 wireless LAN: capacity analysis and protocol enhancement, INFOCOM, 1998.

[26] G. Bianchi, Performance analysis of the IEEE 802.11 distributed coordination function, IEEE Journal of Selected Areas on Communications 18 (3) (2000) 535–547.

[27] S. Sharma, Analysis of 802.11b MAC: A qos, fairness, and performance perspective. Available from: <citeseer.nj.nec.com/sharma03analysis.html>.

[28] H.Y. Hsieh, R. Sivakumar, A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts, in: Proceedings of the ACM MOBICOM, Atlanta, GA, September, 2002.

[29] D.S. Phatak, T. Goff, A novel mechanism for data streaming across multiple IP links for improving throughput and reliability in mobile environments, in: Proceedings of the IEEE INFOCOM 2002, New York, June 2002.
[30] C. Hsin, et al., An analytical study of the effect of lifetime of stored routes on routing control overhead. Available-from: <citeseer.ist.psu.edu/557864.html>.
[31] The Network simulator (NS-2). Available from: <http://www.isi.edu/nsnam/ns>.

Antonios Argyriou is a Ph.D. candidate in the School of Electrical and Computer Engineering, Georgia Institute of Technology. He received his M.S. and Diploma degrees in Electrical and Computer Engineering from Georgia Institute of Technology, and Democritus University of Thrace, Greece, in 2003 and 2001 respectively. His research interests spawn in all aspects of computer networking while specific interests include wireless networks and multimedia communications. He is a student member of IEEE and ACM.

Vijay Madisetti is a professor of Electrical and Computer Engineering at the Georgia Institute of Technology. He splits his time among teaching, research and entrepreneurship. He leads one of the largest research programs in the country in the area of embedded software systems, as a research director of the State of Georgia's Yamacraw Mission. He is also involved in the activities of the Center for Signal and Image Processing (CSIP) and the NSF Packaging Research Center (PRC) as the leader of its Systems on Package (SOP) thrust. He has authored, coauthored, or edited 10 books, including VLSI Digital Signal Processors (IEEE Press, 1995), Quick-Turnaround ASIC Design in VHDL Kluwer, 1996), and Handbook of Digital Signal Processing (CRC Press, 1998). His interests are design, prototyping, and packaging of electronic systems, virtual prototyping, embedded software systems, and computer networks. He obtained his PhD in electrical engineering and computer science from the University of California at Berkeley. He is a member of the IEEE and the Computer Society.