Realizing Load-Balancing in Ad-Hoc Networks with a Transport Layer Protocol

Antonios Argyriou and Vijay Madisetti
School of Electrical & Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332, USA
Email: {anargyr,vkm}@ece.gatech.edu

Abstract— **The advent of wireless ad-hoc networks has provided a cost effective way for exploiting mobile hosts in the absence of infrastructure. However, the full potential that these networks offer for performance enhancement still remains to be exploited. In this paper we exploit the existence of multiple paths between mobile hosts in an ad-hoc network. More specifically, we introduce a new scheme for load-balancing for mobile multi-homed hosts that are part of an ad-hoc network. Our approach differs from previously proposed schemes since it consists an entirely end-to-end approach built on top of the Stream Control Transmission Protocol (SCTP), while the Dynamic Source Routing (DSR) protocol is used for routing. We provide simulation results that prove the efficiency of our approach based on realistic mobility scenarios.**

I. Introduction

Research in the area of wireless ad-hoc networks has focused primarily on the design of efficient routing protocols, primarily because of the mobility of the hosts which also serve as routers. Nevertheless, upper layers protocols and especially at the transport layer, have also proven to be an important bottleneck in achieving good performance, since the majority of them do not consider the special characteristics that ad-hoc networks possess [1]. Their distinctive feature is that they are autonomous and self-organized and in addition they do not necessarily require the use of TCP for communication inside the ad-hoc network. Another important feature of these networks, is the existence of multiple paths between the hosts that change dynamically over time. This is true since the host itself is a router participating in the routing process in the core network.

In this paper we propose a novel load-balancing system for use in ad-hoc networks that is based on a new transport layer protocol, the Stream Control Transmission Protocol (SCTP) [2]. We propose a new algorithm for load-balancing and additionally a number of modifications to the algorithms used by the SCTP protocol. We show that by maintaining multiple active paths at the transport layer, a mobile host can use them efficiently in order to balance its outgoing traffic to the available wireless interfaces. We will use the term LBA-SCTP (Load-Balancing for Ad-hoc networks with SCTP), from now on when we refer to our system.

Related work is mainly focused at routing layer approaches. The authors at [3] for example, propose a system that tries to maintain multiple paths in order to increase the uptime of a connection. In this way the system hides frequent route failures. However, the authors do not address load-balancing for application layer data. Another approach reported in [4] performs a theoretical evaluation of the problem of load-balancing at the routing layer, without providing any implementation. Generally, the related work on this area is restricted.

The rest of this paper is organized as follows: Section II provides an overview of SCTP and DSR since these are the protocols that we will modify. In section III we give an overview of our system while in section IV we describe in detail the proposed modifications to the protocols. Section V provides the simulation results and finally section VI, concludes the paper.

II. Overview of DSR and SCTP

DSR [5], [6] is a simple source routing protocol in which route caching is heavily used. If a 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 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 with a route reply (RREP) packet. In case none of the above holds, the host that received the RREQ re-broadcasts it to its neighborhood. 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. When the source gets this packet removes any route using this link from its cache.

On the other hand, SCTP [2] is a reliable transport protocol that operates on top of a connectionless packet based network such as IP. SCTP is making use of multi-homing, which means that a single association (session) is able to use alternatively anyone of the available interfaces 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 one. Other novel features include the decoupling of reliable delivery from message ordering by the introduction of $streams$. However, congestion control was defined similar to TCP, primarily for being TCP friendly [2].

III. Overview of the proposed system

A load-balancing mechanism cannot be effective unless there are multiple paths between the two nodes that want to complete a data transfer. The current 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). However, this is very useful on our case because the LBA-SCTP system is capable of selecting paths from a number of alternatives. Another key element in our system is that it does not generate any excess probing traffic for discovering active paths, but instead it leaves this process to DSR.

Before we move on with the description of our system we have to clarify a detail related to the MAC layer. We do not consider any MAC layer related problems here. It is possible for example that hosts that content for the medium locally do not allow the exploitation of the hosts for load-balancing even though DSR will provide a bunch of routes. Highly dense topologies will most likely be affected by a problem like this.

A. Disjoint path selection algorithm

Fig. 1 provides a simplified version of the algorithm that identifies 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 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, after 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. the paths overlap). If there is such a case, paths with high RTT are removed from the $LCache$. In this way we only keep the best paths in this cache for future reference. This pruning process goes on until we have a number of non-overlapping equal to the number of available wireless network interfaces in the mobile host. In this way this heuristic provides a fast way for obtaining the best available paths. Even if it is rather obvious, the rationale for judging the paths according to their RTT will be explained later as we delve more into the transport layer issues that arise and 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.

IV. Protocol modifications

The algorithm that we developed earlier, consists only a small part of our system since as we will see, in order to be

```
if $[paths = FIND(LCache, dest)] \neq \emptyset$
    $continue$
elseif $[paths = FIND(RCache, dest)] \neq \emptyset$
    then $ADD(LCache, dest, paths)$
    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.

effective, a number of problems have to be solved first. These problems require a careful cross-layer modification of SCTP, and to a lesser extent of DSR. The optimizations we must en sure that the effects of concurrent data transmission and frequent link breakages are masked from the application layer. In order to achieve that, we modified three functional parts of SCTP. More specifically these are: 1) The path heart-beating mechanism, 2) the retransmission algorithm, 3) and the data sending process. In the case of DSR the only requirement is to provide the available routes ($RCache$) at the transport layer.

A. Efficient data stripping across multiple links

Assume that N is the number of total wireless interfaces used by the mobile host (practically this is usually between 2-3). If we assume that each outgoing packet has the same size, the fraction of data that must be sent in each link in order to avoid spurious retransmissions, according to [7], will have to be:

$$f = \frac{b_i}{b_{total}} \tag{1}$$

Basically this result verifies our intuition that we should sent data to each path according to the available bandwidth. If a transport layer connection sends more data than this equation specifies in the corresponding link, then data will not be delivered soon enough, and the limit of 3 duplicate acknowledgements [2] at the sender will soon be reached and the sender will fast retransmit segments that are still in transit. We follow this simple rule for stripping data to outgoing interfaces from now on. Note that before the data-stripping process starts the algorithm in Fig. 1 is executed.

B. Avoiding RTO timeouts

Despite the fact that the algorithm in 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 sequence number 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 initiate the fast retransmission process at the sender [2]. In this way the first transmission is useless and in addition the sender throttles back its congestion window [7]. This effectively means that the whole aggregate connection will be limited by the slowest link present.

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:

$$RTO_i = SRTT_i + 4 * RTTVAR_i \tag{2}$$

while $SRTT$ and $RTTVAR$ are:

$$SRTT_i = (1 - RTO.\alpha) * SRTT + RTO.\alpha * R' \tag{3}$$

$$\begin{aligned} RTTVAR_i = (1 - RTO.\beta) * RTTVAR_{i-1} \\ + RTO.\beta * |SRTT - R'| \end{aligned} \tag{4}$$

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

$$\overline{RTT} = \sum_{i=1}^{N} \overline{RTT}_i \times \frac{n_i}{n_{total}} \tag{5}$$

$$\overline{RTT_{DEV}} = \sum_{i=1}^{N} (RTT_i - \overline{RTT}_i) \times \frac{n_i}{n_{total}} \tag{6}$$

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

$$RTT_i < \overline{RTT} + 4 * \overline{RTT_{DEV}} \tag{7}$$

If this constrain is not satisfied then the specific route is not pushed to the load-balancing cache ($LCache$). On the other hand, if the constraint is satisfied, the route is pushed to the $LCache$. We saw earlier that when there is a need to send a new packet, the algorithm in Fig. 1 will be executed comparing the RTT between all the paths for a given destination. Note that we do not remove a route from the DSR route cache even if it does not satisfy eqn. 7, since RTT might soon change and make the route usable.

C. Enhancing the heartbeating mechanism

According to the SCTP RFC [2], each endpoint of an SCTP association can configure a number of addresses. The protocol sends Heartbeat messages that monitor the status of all these remote addresses. Note however, that SCTP cannot monitor the actual path status, but only the destination address. However, when source routing is used, as it is the case with our system (DSR), the nodes have exact knowledge of the specific path that a packet had taken/will take. This means that now the heartbeat mechanism is transformed in actual path monitoring mechanism.

Furhtermore, our system assigns the best paths to the main SCTP data flows while, for the time being, the only packets sent to some of the secondary paths that do not meet eqn. 7, are Heartbeat messages. By default, SCTP will start sending Heartbeat messages periodically (every RTO) to every destination address that belongs to the association. Therefore, we redundantly insert some paths to the SCTP association, that are not currently used, but they are implicitly monitored. Since the heart-beating mechanism involves a Heartbeat ACK packet, we can have explicit feedback about the status of a specific path. In this way we can monitor the RTT for paths that are currently not used but they make soon have lower RTT. This approach in our system, constitutes a crucial difference with the current DSR-based load-balancing systems like the ones reported in [3] and [4], since in these cases, DSR informs with a RERR message only if the primary path was down. However, in our LBA-SCTP optimized system, since we maintain a small number of additional paths used for load-balancing, we get status information every RTO. It is not necessary for a path to go completely down/up in order to remove/add it in the $LCache$ of usable paths. This is the result of our novel mechanism that keeps track of the real path status at the transport layer.

D. Avoiding route discovery delay

According to the DSR protocol, when a currently used route fails, DSR starts the route discovery process in order to find a new route to the destination. As we said, in the previous subsection, it is possible this process may last more than the RTO timer at the sender [1]. This will result in a timeout, packet retransmission and reduction of the congestion window. Even after a new route has been found, an SCTP or TCP 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.

However, in LBA-SCTP we saw that we practically move ”one layer up” the actual route selection, that is performed at the transport layer. By maintaining multiple routes in LBA-SCTP, the transport layer has one more degree of flexibility concerning the actual route for its packet. The advantage of this approach in this case, is that we avoid the lengthy route discovery process in case of a route failure. Even if we follow the approach that freezes the sender's state [8], the freezed state will not correspond after a while to the new path status but to the old one. However, SCTP maintains congestion control parameters for each path or source-destination address pair, allowing us thus to maintain a distinct state for each path, which in conjunction with heartbeating provides up-to-date information for them.

E. Overcoming transient packet reordering

After a route failure, and in the absence of alternative paths, the sender initiates a route discovery process, and upon reception of the new route, the packets of the sender are sent to the new route. A problem that might show up here, is the out-of-order delivery of the sender packets due to the asymmetry of the paths, buffering delays etc. Similarly to section IV-B, this side-effect will trigger duplicate ACK (SACK for SCTP) transmission from the part of the receiver which will inevitably result in an invocation of the congestion control algorithm at the sender with all its bad consequences. This is a major problem with multi-path routing protocols such as TORA [9].

The problem of packet reordering can be appropriately tackled by careful cross layer optimization in the context of SCTP and DSR. We are capable of doing this because we have exact knowledge of the time the sender switches to an alternative path by DSR. LBA-SCTP is notified with an upcall from the DSR when a route fails, so that LBA-SCTP switches to an alternative path from the $LCache$. We force LBA-SCTP to also do that by itself as we earlier said after the $2RTO$ timer expires on the primary path.

This is clearly a transient behavior which LBA-SCTP enters when a path fails and a new one must be used. Fig. 2 presents how the LBA-SCTP state machine behaves so that it overcomes the transient packet reordering.

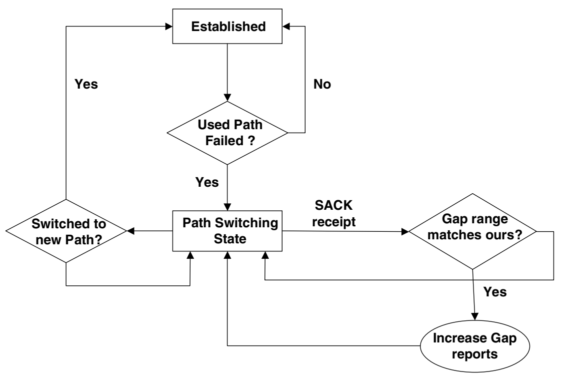


Fig. 2. Modified SCTP state machine.

In our implementation, the LBA-SCTP sender maintains two variables for each local IP address, that keep the lowest and highest TSN sent during the last $cwnd$ to the receiver. When the receiver replies with a SACK containing a Gap report for TSNs that do not belong to the range recorder at the sender, the sender does not increase the Gap Ack reports and processes the SACK chunk as a normal SACK that acknowledges the outstanding data for this transport address only. The rationale behind this is that the chunks that the receiver is requesting were not send by its address were sent but by the new address. In the meantime, the receiver monitors its $CumTSNack$ [1] and when data are received from any address can fill the gap then the receiver stops sending gap reports with SACK messages. However, in order to take into consideration the fact that packets might be lost, the sender retransmits after five duplicate acknowledgements from the receiver. By using this dual route failure detection mechanism we are sure that the time that a specific wireless interface at the sender stays idle is minimized.

V. PERFORMANCE EVALUATION

In this section we provide simulation results after we implemented all the previously mentioned algorithms and modifications. The Ns-2 network simulator [10] was used for all our experiments. We used 802.11 as the MAC level protocol, DSR which is integral part of the NS-2 simulator, and the SCTP implementation that is available at [11].

We defined a 600mX600m area for host movements. The simulation time is 200 seconds 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 [5]. In a movement scenario, each host starts by being initially stationary for a duration of $pause$ time (seconds). After this time frame, the model requires it to select a random destination in the 600mX600m 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. The majority of the results, unless otherwise specified, corresponds to host speed of 20 m/s. Backlogged FTP sources were used, creating randomly 20 source-destination pairs out of a total of 50 nodes in a scenario.

A. Simulation results for LBA-SCTP

In our first experiment we limited the maximum data rate per connection to 33Kbps, in order to see the effects of our system at low bit-rates. Fig. 3 shows results for one specific data flow between two hosts. Three plots are shown: the ideal average aggregate throughput, aggregate throughput with LBA-SCTP/DSR, and throughput for unmodified SCTP/DSR (one link). It is clear that LBA-SCTP 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

[1]Cumulative TSN Ack Point: Equivalent to TCP's largest sequence number received

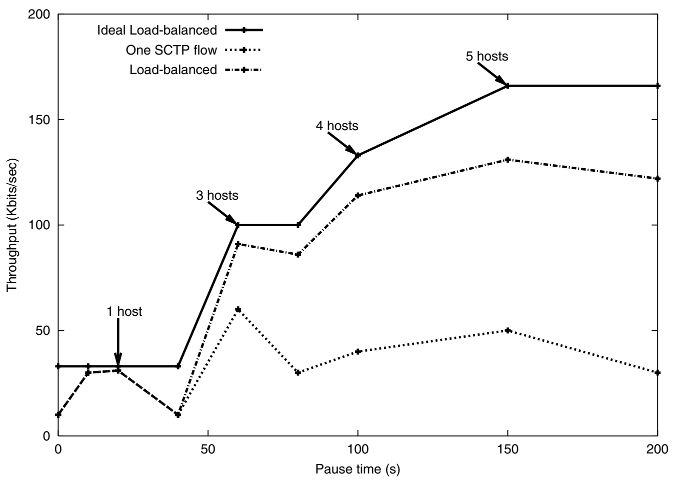
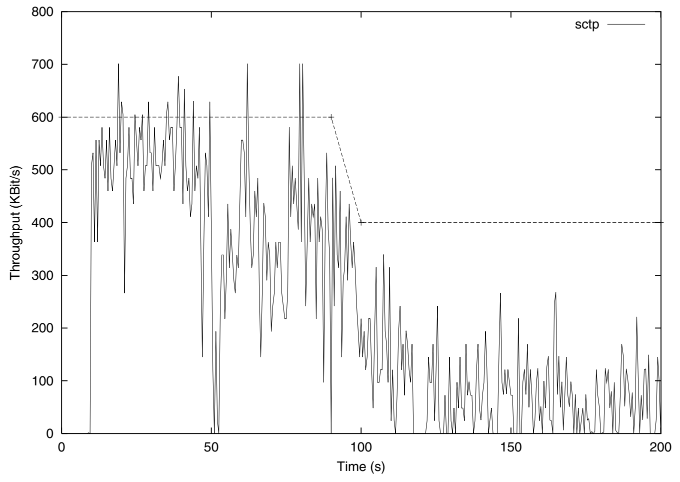
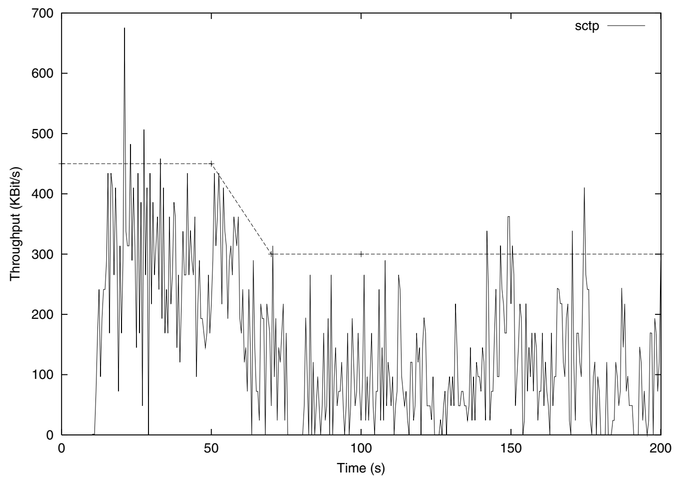


Fig. 3. LBA-SCTP aggregate throughput. The mobile host is equipped with five wireless interfaces.



(a) Local contention after 100 sec prevents the host from using the full aggregate bandwidth.



(b) LBA-SCTP takes advantage of the low loaded medium for load-balancing.

Fig. 4. Instantaneous throughput for two different LBA-SCTP flows.

while it is quite increased for low mobility cases. The arrows in Fig. 3 indicate the average number of available links with neighbor hosts that can be used for concurrent transmission. For example for high mobility scenarios (pause time between 0-40 sec), 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.

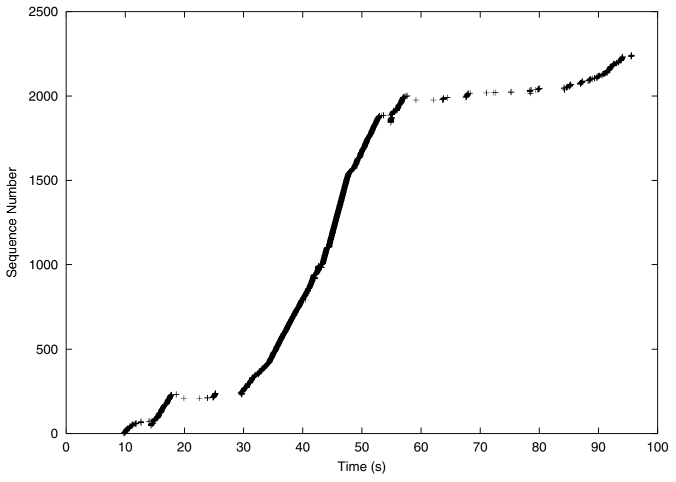
Fig. 4(a) shows the instantaneous aggregate throughput for one flow of one simulation round of 200 seconds. In addition we show, the available aggregate bandwidth according to the number of available links as before. For a significant amount of time the host captures nearly the full amount of the available bandwidth (3 hosts $\times$ 200 Kbps) . After time 100 sec, 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 a small fraction 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. 4(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. 4(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. Still though, before time 70 sec, the host enjoyed better throughput since one more inactive host was present.

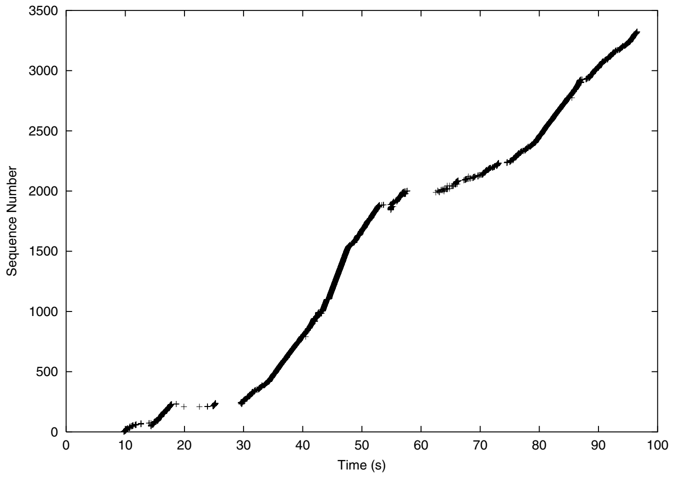
So the important conclusion is the following: More "dense" topologies result in the existence of more routes in the route cache allowing greater flexibility in the route selection process. However, when at the same time the number of flows are increased from these hosts, contention is increased and so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

Finally Fig. 5 presents sequence number progression at the receiver for the same scenario, for a typical flow between two hosts. We can see now that LBA-SCTP does not suffer from the timeout in Fig. 5(b) as opposed to the original SCTP version in Fig. 5(a). And the reason is that at time 60 sec, after $2RTO$ the LBA-SCTP sender switches to an existing additional path.

We also measured the packet delivery fraction (PDF) for the same experiment as above, but due to lack of space no results are presented. The important result from this measurements is that the packet delivery ratio was found to be higher in the case of LBA-SCTP, and this is attributed to the avoidance of



(a) SCTP experiences an RTO timeout.



(b) LBA-SCTP avoids RTO timeout.

Fig. 5. Sequence number progression. Load-balancing is performed over two links in case (b).

transient packet losses in the way we described earlier.

VI. CONCLUSIONS

In this paper we presented a system for transport layer load-balancing in ad-hoc networks. The proposed algorithms were incorporated to the SCTP protocol, along with a number of modifications to the vanilla SCTP implementation. Our system is completed with the use of DSR as a routing protocol. We showed with simulation results, that the proposed system is capable of efficiently using the available links for aggregating its outgoing traffic. The number of necessary modifications to the protocols is restricted, making thus our solution a good candidate for practical applications in ad-hoc networks.

REFERENCES

[1] J. Monks, P. Sinha, and V. Bharghavan, “Limitations of TCP-ELFN for Ad Hoc Networks,” in *MOMUC*, Tokyo, Japan, October 2000.

[2] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, “Stream Control Transmission Protocol,” RFC 2960, October 2000.

[3] R. Leung, J. Liu, E. Poon, C. Chan, and B. Li, “MP-DSR: A QoS-aware Multi-path Dynamic Source Routing Protocol for Wireless Ad-hoc Networks,” Available from url =”citeseer.nj.nec.com/557864.html”.

[4] L. Zhang *et al.*, “Load Balancing of Multipath Source Routing in Ad-Hoc Networks,” in *International Conference on Communications (ICC) 2002*.

[5] D. B. Johnson and D. A. Maltz, “Dynamic Source Routing in Ad hoc Wireless Networks,” in *Mobile Computing*, T. Imielinski and H. Korth, Eds. Kluwer Academic Publishers, 1996, ch. 5, p. pages 153181.

[6] C. Perkins, E. Royer, S. Das, and M. Marina, “Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks,” *IEEE Personal Communications*, February 2001.

[7] D. S. Phatak and T. Goff, “A Novel Mechanism for Data Streaming Across Multiple IP Links for Improving Throughput and Reliability in Mobile Environments,” in *IEEE INFOCOM*, vol. 2, 2002, pp. 73–781.

[8] G. Holland and N. Vaidya, “Analysis of TCP Performance over Mobile Ad Hoc Networks,” in *IEEE/ACM Mobicom*, 1999.

[9] J. Liu and S. Singh, “ATCP: TCP for Mobile Ad-Hoc Networks,” *IEEE Journal in Selected Areas in Communications*, 2001.

[10] S. McCanne and S. Floyd, “ns Network Simulator,” http://www.isi.edu/nsnam/ns/.

[11] A. L. Caro *et al.*, “SCTP module for ns-2,” http://www.pel.udel.edu.