TPA: a Transport Protocol for Ad hoc Networks

G. Anastasi[*], E. Ancillotti[*§], M. Conti[§], A. Passarella[*]

Pervasive Computing & Networking Laboratory (PerLab)

[*]*Dept. of Information Engineering, University of Pisa, Via Diotisalvi 2 - 56122 Pisa, Italy*
{giuseppe.anastasi, andrea.passarella, emilio.ancillotti}@iet.unipi.it

[§]*CNR-IIT Institute, Via G. Moruzzi, 1 - 56124 PISA, Italy*
marco.conti @iit.cnr.it

Abstract

Several previous works have shown that TCP exhibits poor performance in Mobile Ad Hoc Networks (MANETs). The ultimate reason for this is that MANETs behave in a significantly different way from traditional wired networks, like the Internet, for which TCP was originally designed. In this paper we propose a novel transport protocol – named TPA – specifically tailored to the characteristics of the MANET environment. It is based on a completely new congestion control mechanism, and designed in such a way to minimize the number of useless transmissions and, hence, power consumption. Furthermore, it is able to manage efficiently route changes and route failures. We evaluated the TPA protocol in a static scenario where TCP exhibits good performance. Simulation results show that, even in such a scenario, TPA significantly outperforms TCP.

1. Introduction

In the last years the research activities on MANETs have pointed out that the TCP behavior in a multi-hop ad hoc network is far from ideal. The ultimate reason for this is that MANETs behave in a significantly different way with respect to traditional wired networks, like the Internet, for which the TCP protocol was originally conceived [14].

To improve the performance of the TCP protocol in MANETs several proposals have been presented [1, 3, 5, 6, 8, 11]. Almost all these proposals are modified versions of the legacy TCP protocol. However, we think that it is more fruitful to think in terms of a new transport protocol optimized for MANETs rather than adapting TCP to the ad hoc environment. The motivations for a novel transport protocol, instead of a modified TCP, are thoroughly discussed in [14]. The authors in [4] take an approach similar to ours.

In this paper we propose a transport protocol, TPA (Transport Protocol for Ad hoc networks), specifically tailored to the characteristics of the MANET environment. It provides a reliable, connection-oriented type of service and includes mechanisms to manage route failures and route changes that may arise due to nodes' mobility. In addition, the congestion control mechanism is completely re-designed with respect to the legacy TCP. Finally, TPA implements a novel retransmission policy to reduce the number of useless retransmissions and, hence, the power consumption.

The rest of the paper is organized as follows. Section 2 describes the TPA protocol. Section 3 is devoted to the TPA performance evaluation. Section 4 concludes the paper.

2. TPA Protocol Description

The TPA protocol provides a reliable, connection-oriented type of service. The set up and tear down phases are similar to the corresponding phases in the TCP protocol, and are thus omitted for the sake of space. In the following we will briefly describe the data transfer phase (see [15] for details) .

2.1. Data Transfer

TPA is based on a sliding-window scheme where the window size varies dynamically according to the flow control and congestion control algorithms. The congestion control mechanism is described in Section

Work carried out under the financial support of the FET IST Mobile MAN Project (IST-2001-38113) and the Italian Ministry for Education and Scientific Research (MIUR) in the framework of the FIRB Projects VICOM and PERF.

2.4, while the flow control mechanism is similar to the corresponding TCP mechanism [10], and is omitted.

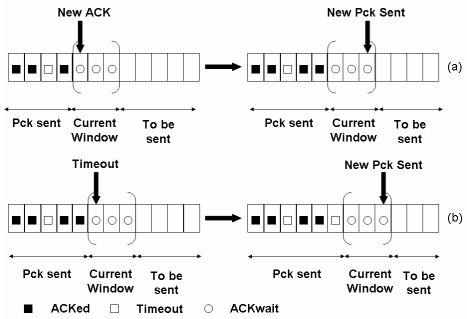

Figure 1. ACK reception (a), and timeout expirations (b).

TPA tries to minimize the number of (re)transmissions in order to save energy. To this end, packets to be transmitted are managed in blocks, with a block consisting of *K* packets[1]. The source TPA grabs a number of bytes - corresponding to K TPA packets - from the transmit buffer[2], encapsulates these bytes into TPA packets, and tries to transmit them reliably to the destination. Only when all packets belonging to a block have been acknowledged, TPA takes care to manage the next block. Each packet header includes a *sequence number* field that identifies the block to which the packet belongs, and a *data_bitmap* field consisting of *K* bits to identify the position of the packet within the block. The TPA header also includes two fields for piggybacking ACKs into data packets: *acknowledgement number* and *ack_bitmap*. The *acknowledgement number* identifies the block containing the packet(s) to be acknowledged, while a bit set in the *ack_bitmap* indicates that the corresponding packet within the block has been received correctly by the destination. Of course, it is possible to acknowledge more than one packet by setting the corresponding bits in the bitmap (a single ACK contains information for all the packets within the block).

Packet transmissions are handled as follows. Whenever sending a packet, the source TPA sets a timer and waits for the related ACK from the destination. Upon receiving an ACK for an outstanding packet the source TPA performs the following steps: i) derives the new window size according to the congestion and flow control algorithms (see below); ii) computes how many packets can be sent according to the new window size; and iii) sends next packets in the block (see Figure 1a). On the other hand, whenever a timeout related to a packet in the current window expires, the source TPA marks the packet as "timed out" and executes steps i)-iii) as above, just as in the case the packet was acknowledged (see Figure 1b).

In other words, TPA performs a transmission round during which it sends all packets within the block, without retransmitting timed-out packets. Then, the sender performs a second round for retransmitting timed-out packets, which are said to form a "retransmission stream" (see Figure 2). In the second round the sender performs steps i)-iii) described above with reference to the retransmission stream instead of the original block. This procedure is repeated until all packets within the original block have been acknowledged by the destination. If an ACK is received for a packet belonging to the retransmission stream, that packet is immediately dropped from the stream.

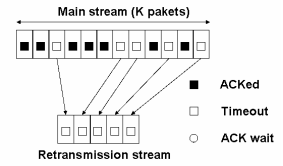

Figure 2. Retransmission Stream.

The proposed scheme has several advantages with respect to the retransmission scheme used in TCP. First, the probability of useless transmissions is reduced since packets for which the ACK is not received before the timeout expiration are not retransmitted immediately (as in the TCP protocol) but in the next transmission round. This is particularly important in MANETs where nodes are highly mobile and, thus, the timeout value might not reflect the current Round Trip Time (RTT) of the connection (see also Section 2.2). It should also be observed that the longer waiting time in the TPA protocol does not result in throughput degradation, since during this time interval the sender transmits other packets. Second, TPA is resilient against ACK losses because a single ACK is sufficient to notify the sender about all missed packets in the current block. Third, the sender does not suffer from out-of-order arrivals of packets. This implies that TPA can operate efficiently also in MANETs using multi-path forwarding [2].

2.2. Route Failure Management

Like many other solutions [1, 3, 5, 6], TPA can exploit, if available, the *Explicit Link Failure Notification* (ELFN) service provided by the network-layer for detecting route failures.

Upon receiving an ELFN, the source TPA enters a *freeze* state where the transmission window size is limited to one packet. To limit the number of packets sent when there is no available route, while in the freeze

1 TPA packets have the same size of TCP segments.

[2] A block may include less than *K* packets if the buffer does contain a sufficient number of bytes.

state the value of the retransmission timer is doubled after each timer expiration [15].

However, even if the underlying layer does not provide the ELFN service, the sender TPA is still able to detect route failures as it experiences a number of consecutive timeout. Specifically, the sender TPA assumes that a route failure has occurred whenever it detects th_{ROUTE} consecutive timeouts. In this case it enters the *freeze* state. While in the *freeze* state, the TPA sender behaves as described above. Obviously, th_{ROUTE} is a protocol parameter that needs to be set appropriately.

We assume that the network layer does not provide route re-establishment notifications. Therefore, TPA realizes that the route has been re-established as soon as it receives an ACK for the latest packet sent. Upon reception of such an ACK, TPA i) leaves the *freeze* state; ii) sets the congestion window to the maximum value *CWNDmax*; and iii) starts sending new packets [15]. On the other hand, if route re-establishment message are available, the TPA behavior can be further optimized. Specifically, in the freeze state TPA can refrains from transmitting any packet, waiting for a route re-establishment message.

2.3. Route Change Management

Similarly to TCP, TPA estimates the connection RTT, and uses this estimate to set the Retransmission Timeout (RTO). Both parameters are derived in the same way as in the TCP protocol, i.e.:

$$ERTT_{rtt}(n) = g \times RTT(n) + (1-g) \times ERTT_{rtt}(n-1)$$
$$DEV_{rtt}(n) = h \times |RTT(n) - ERTT_{rtt}(n)| \times (1-h) \times DEV_{rtt}(n-1)$$
$$RTO(n) = ERTT_{rtt}(n) + 4 \times DEV_{rtt}(n)$$

where: i) $ERTT_{rtt}(n)$ and $DEV_{rtt}(n)$ are the average value and standard deviation of the RTT estimated at the n^{th} step, respectively; ii) *RTT(n)* denotes the n^{th} RTT sample; iii) *RTO(n)* is the retransmission timeout computed at the n^{th} step; and iv) g and h ($0 < g, h < 1$) are real parameters (see [10] for details).

Whenever a route change occurs, the new path may differ from the previous one in terms of number of hops. This means that, after a route change, packets may experience a variation in the RTT and the retransmission timeout might be no longer appropriate for the new path. To avoid possible re-transmissions, the TPA protocol must detect route changes as soon as they occur, and modify the RTT estimation method to achieve quickly a reliable estimate for the new RTT. In practice, TPA detects that a route change has occurred either i) when a new route becomes available after a route failure; or ii) when th_{RC} consecutive samples of the RTT are found to be external to the interval $[ERTT_{rtt} - DEV_{rtt}, ERTT_{rtt} + DEV_{rtt}]$. Upon detecting a route change, TPA replaces the g and h values in the ERTT and DEV estimators by greater values (*g1* and *h1*) so that the new RTT estimates are heavily influenced by the new RTT samples. This allows to achieve a reliable estimate of the new RTT immediately after the route change has been detected. Finally, after n_{RC} updates of the estimated RTT, the parameter values are restored to the normal g and h values.

2.4. Congestion Control Mechanism

Congestions due to link-layer contentions manifest themselves at the transport layer in two different ways. An intermediate node may fail in relaying data packets to its neighboring nodes and, thus, it sends an ELFN back to the sender node (provided that this service is supported by the network layer). This case, throughout referred to as *data inhibition*, cannot be distinguished by the sender TPA from a real route failure. On the other hand, an intermediate node may fail in relaying ACK packets. In this case, throughout referred to as *ACK inhibition*, the ELFN (if available) is received by the destination node (i.e., the node that sent the ACK), while the source node (i.e., the node sending data packets) only experiences consecutive timeouts. Whenever the sender TPA detects th_{CONG} consecutive timeout expirations it assumes that an ACK inhibition has occurred, and enters the *congested state*. The source TPA leaves the congested state as soon as it receives th_{ACK} consecutive ACKs from the destination.

If the network layer does not support the ELFN service, the only way to detect both data and ACK inhibitions is by detecting consecutive timeouts at the sender. Congestions and route failures are no longer distinguishable. Hence, th_{CONG} and th_{ROUTE} collapse in the same parameter, and the *freeze* and the *congested* states collapse in the same state.

The TPA congestion control mechanism is window-based as in the TCP protocol. However, in TPA the maximum congestion window size (*CWNDmax*) is very small (in the order of 2-3 TPA packets) and, hence, the maximum and minimum values are very close. Therefore, the TPA congestion control algorithm is very simple. In normal operating conditions, i.e., when TPA is not in the *congested* state, the congestion window is set to the maximum value, *CWNDmax*. When TPA enters the *congested* state, the congestion window is reduced to 1 to allow congestion to disappear.

3. Simulation Analysis

In this section we compare, by simulation, the performance of TCP and TPA. To this end, we developed

a custom simulation model that extends the model used in [7]. Our simulator includes the IEEE 802.11 MAC protocol and physical channel model, the DSR routing protocol [12], and the transport protocol (TCP or TPA). In this model we assumed the same physical channel model used in the ns-2 simulation tool (see Table 1). The carrier sensing range is thus more than twice the transmission range, while the interference range is the same as the carrier sensing range.

Table1. Physical channel model.

Parameter	Value
Bit rate	2 Mbps
Transmission range	376 m
Interference range	676 m
Carrier sensing range	676 m

Table2. Operational Parameters.

Parameter	Value
Distance between node	300 m
Packet Size (TCP/TPA)	512 Bytes
CBR Packet Size (UDP)	512 Bytes
th_{ROUTE} (TPA)	3
th_{ACK} (TPA)	1
Block Size (TPA)	12

To exercise the TCP protocol in a favorable environment we considered a static scenario (i.e., absence of node mobility). Furthermore, we assumed that the network layer provides neither the ELFN nor the route re-establishment services.

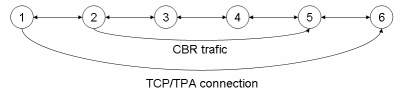

Figure 3. Network Configuration.

In our experiments we considered the string topology depicted in Figure 3, where the distance between consecutive nodes is 300 m (the other operation parameters are shown in Table 2). Therefore, each node is within the transmission range of only adjacent nodes, and the carrier sensing and interference ranges span two hops (for instance, when node 4 in Figure 3 is transmitting, nodes 2 and 6 can hear its transmission). According to the motivations expressed in several previous works (see [14] and references therein), in our experiments we considered a TCP/TPA connection that spans a limited number of hops. We assumed that node 1 is sending ftp-like traffic to node 6. In addition, to investigate the effects of background traffic on the performance of the TCP/TPA connection, we also considered a CBR (Continuous Bit Rate) session where node 2 sends periodically (UDP) packets to node 5. We compared the performance of TCP and TPA both in terms of throughput achieved by the destination node at the application layer, and percentage of retransmissions, i.e., percentage of packets retransmitted by the TCP/TPA sender. Since (re-)transmissions consume energy (both at the sender and intermediate nodes) the percentage of retransmissions can be regarded as an energy-efficiency index.

3.1. Simulation Results[3]

As a preliminary step in our analysis, we exercised TCP and TPA in the network scenario described above to determine the optimal maximum window size. In these experiments we did not consider any background traffic. The results obtained are summarized in Figure 4. It clearly appears that the optimal window size for both protocols is 2. For the TCP protocol this result is aligned with the analysis in [13,9]. In this scenario both protocols exhibit similar performance both in terms of throughput and retransmissions. The reason is that in a static MANET without any interfering traffic the TCP protocol with limited maximum window size exhibits good performance [9].

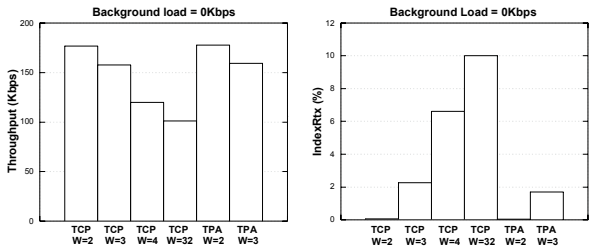

Figure 4. Throughput (left) and retransmission index (right) vs. window size.

In the following sections we will focus on the performance of TPA and TCP with maximum window size equal to 2. However, for completeness, in plots we also show curves related to maximum window sizes greater than 2. We will investigate how the performance of both protocols are influenced by factors such as interfering traffic produced by others sessions, latency for discovery a new route whenever a route failure occurs, and presence of a selfish node along the connection path.

3.1.1. Impact of the Background Traffic. To investigate the effects of the background traffic we ran a set of experiments with the CBR session active. In this set

[3] The results shown in this section were obtained by using the independent replication method with a confidence level of 95%. The confidence interval is always in the order of some percents.

of experiments we assumed a route-recovery latency equal to 0, i.e., we assumed that a new route is immediately found after a route failure. The results in Figure 5 show that TPA tends to outperform TCP as the background load increases. This is especially highlighted by the retransmission index. Even with a maximum window size equal to 2, when the background load reaches 150 Kbps, the percentage of retransmitted packets is halved by TPA (6% vs. 12%). In addition, TPA provides a higher throughput. In conclusion, TPA provides a higher throughput than TCP, while consuming roughly half of the energy spent by TCP in retransmissions.

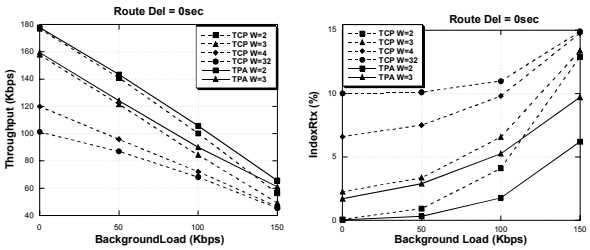

Figure 5. Throughput (left) and retransmission index (right) vs. Background Traffic.

3.1.2. Impact of the Route Discovery Latency. In static MANETs route failures occur due to link-layer contentions. Whenever a route failure is detected, the routing protocol tries to find an alternative route. The discovery of the new route may take some time. In our experiments a fixed delay parameter, *RouteDel*, represents the amount of time during which no route to the destination is available after a route failure. Plots in Figure 6 show the effects of increasing the route discovery latency. Plots in the top show that, in the absence of background traffic, the route discovery latency has no meaningful effect. TPA only provides small improvements over TCP with window size equal to 2. This is, because with a window size equal to 2, the number of route failures is close to zero.

The situation is quite different when there is interfering traffic. Plots in the bottom of Figure 6 show that, even with a moderate background traffic (i.e., 50 Kbps), TPA largely outperforms TCP both in terms of throughput and percentage of retransmissions. Specifically, when the route-discovery latency is 1 second, the throughputs achieved by TPA and TCP are 124.9 and 71.8 Kbps, respectively. In addition, the TCP retransmission index is 8.9 %, while the TPA retransmission index is still close to 1%. These performance differences stem from the fact that TPA leverages buffer functionalities that are implemented in typical MANET network protocols. MANET routing protocols (e.g., DSR) usually buffer packets generated by the sender while a new route is being searched. When the congestion level in the MANET increases, the number of packets required to find a new route increases as well. In this case, TCP keeps retransmitting always the same packet until the new route is found, since its congestion window size is stuck at 1. On the contrary, TPA transmits successive packets in the main or retransmission streams. These packets are buffered at the sending-node network layer, and thus they are immediately delivered once a new route is found.

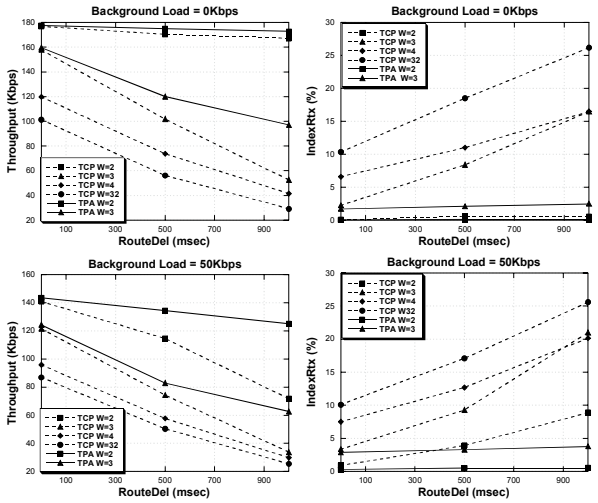

Figure 6.Throughput and retransmission index vs. Route Discovery Latency, without (top) and with (bottom) Background Traffic.

We performed additional experiments with increasing background traffic and found that the difference between TPA and TCP performance becomes larger and larger as the background traffic grows up (results are not shown for the sake of space).

3.1.3. Impact of node selfishness. In this section we investigate the impact of node selfishness. MANETs rely on the assumption that intermediate nodes are willing to forward data traffic originated by other nodes towards the final destination. However, an intermediate node might not cooperate either because it is selfish or because it has limited energetic resources (that are deserved to local traffic). In our simulations we modeled the behavior of a selfish node by assuming that it does not forward (i.e, discards) a packet generated by another node with probability p (p defines the degree of node selfishness). Specifically, in our experiments, we assumed that the selfish node is node 3 (see Fig. 3).

Figure 7 shows the performance of TPA and TCP for different values of node selfishness. Even in this

case, the advantage of using TPA instead of TCP resides both in the lower number of retransmissions, and in the higher throughput. At a selfishness level of 10%, TPA requires around 34% less retransmissions than the TCP, and the TPA throughput is 7% higher than the TCP throughput. At a selfishness level of 50%, the TPA throughput is 150% higher than the TCP throughput, and the TPA retransmission index is 20% lower.

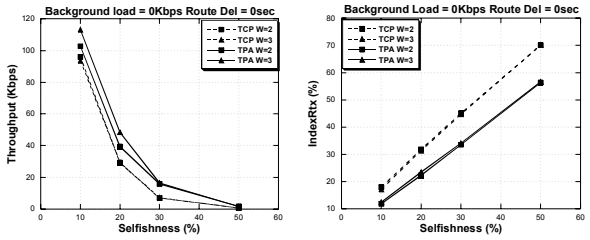

Figure 7. Throughput (left) and retransmission index (right) vs. node selfishness.

4. Conclusions

In this paper we have proposed a novel transport protocol for ad hoc networks, TPA, specifically tailored to the characteristics of the MANET environment. This proposal is motivated by the evidence that the TCP protocol exhibits poor performance in MANETs. The ultimate reason for this is that MANETs behave in a significantly different way with respect to traditional wired networks, like the Internet, for which the TCP protocol was originally conceived.
We have compared, by simulation, the performance of TPA and TCP (with limited maximum window size) in a static scenario. The results obtained show that TPA outperforms TCP in all operating conditions. Specifically, the TPA protocol is able to conserve energy by avoiding many useless transmissions, while providing, at least, the same throughput provided by TCP. The analysis of TPA performance in a mobile environment has been left for further study.

5. References

[1] K. Chandran, S. Raghunathan, S. Venkatesan, R. Prakash, "A Feedback Based Scheme for Improving TCP Performance in Ad-Hoc Wireless Networks", *Proceedings of ICDCS '98*, pp. 472-479.

[2] V.D. Park and M.S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks", *Proceedings of IEEE INFOCOM '97*, Kobe (Japan), 1997.

[3] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks", *Wireless Networks*, Vol.8, pp. 275-288, 2002.

[4] K. Sundaresan, V. Anantharaman, H.-Y. Hsieh, and R. Sivakumar, "Atp: A Reliable Transport Protocol for Ad hoc Networks," *Proc. ACM Symposium on Mobile Ad Hoc Network and Computing (MobiHoc 2003)*, Annapolis (Maryland), June 2003.

[5] J. Liu and S. Singh, "ATCP: TCP for Mobile Ad Hoc Networks", *IEEE J-SAC*, Vol. 10, No. 7, July 2001.

[6] D. Sun and H. Man, "ENIC - An Improved Reliable Transport Scheme for Mobile Ad Hoc Networks", *Proceedings of the IEEE Globecom Conference*, 2001.

[7] F. Calì, M. Conti and E. Gregori, "IEEE 802.11 wireless LAN: capacity analysis and protocol enhancement", *IEEE Transanction on Networking*, Vol. 8, N. 6, pp. 785-799, December 2000.

[8] F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response", *proceedings of MobiHoc 2002*.

[9] S. Xu, T. Saadawi, "Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks", *Comp. Networks* 38 (2002), pp. 531.548.

[10] W.R. Stevens, "TCP/IP Illustrated", Vol. 1, Addison Wesley, 1994.

[11] D. Kim, C. Toh, Y. Choi, "TCP-Bus: Improving TCP Performance in Wireless Ad-Hoc Networks", *proceedings of ICC*, 2000.

[12] D. B. Johnson, D. A. Maltz and Y.-C. Hu, "The dynamic source routing protocol for mobile ad hoc networks (DSR)", *Internet Draft IETF MANET WG*, July 2004.

[13] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang and M. Gerla, "The Impact of Multihop Wireless Channel on TCP Throughput and Loss", *Proceedings of IEEE INFOCOM 2003*, San Francisco (California), March 30.April 3, 2003.

[14] G. Anastasi, A. Passarella, "Towards a Novel Transport Protocol for Ad Hoc Networks", Proc. *IFIP Int. Conference on Personal Wireless Communications (PWC 2003)*, Sept. 23-25, 2003, Venice (Italy), *LNCS*, N. 2775.

[15] G. Anastasi, E. Ancillotti, M. Conti, A. Passarella, "TPA: A Transport Protocol for Ad hoc Networks: Extended Version", http://www.iet.unipi.it/~anastasi/papers/tpa.pdf/.