Towards a Novel Transport Protocol for Ad Hoc Networks*

G. Anastasi and A. Passarella

University of Pisa, Dept. of Information Engineering
Via Diotisalvi 2 - 56122 Pisa, Italy
{g.anastasi,a.passarella}@iet.unipi.it

Abstract. The TCP protocol exhibits poor performance in multi-hop 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 the TCP protocol was originally designed. In particular, route failures and route changes due to node mobility may be frequent events in MANETs. Furthermore, congestion phenomena in MANETs are essentially different from traditional wired networks. In this paper we propose a novel transport protocol for MANETs. Unlike other proposals, our protocol is not a modification of the TCP but is specifically tailored to the characteristics of the MANET environment. It is able to manage efficiently route changes and route failures. Furthermore, it includes a completely re-designed congestion control mechanism. Finally, it is designed in such a way to reduce as much as possible the number of useless retransmissions. This is extremely important since retransmissions consume energy.

Keywords: Ad Hoc Networks, Mobility, Transport Protocols, TCP.

1 Introduction

The proliferation of portable computers (notebooks, palm tops, PDAs, smart phones) and the development of wireless technologies has spurred the interest towards Mobile Ad Hoc Networks (MANETs). Thanks to their self-organizing nature, MANETs are suitable for applications that must be deployed in high dynamic scenarios (e.g., peer to peer communications within a campus, support in rescue operations, etc.).

In the last years the research activities in the field of MANETs have mainly focused on routing protocols [1,2]. In addition, several papers have pointed out that the TCP behavior in a multi-hop ad hoc network is far from ideal. Many aspects contribute to this non-ideal behavior, some of which are discussed in Sect. 2. To improve the performance of the TCP protocol in multi-hop ad hoc networks several proposals have been presented [3,4,5,6,7,8]. To the best of our

* This work was partially funded by the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2001-38113 MOBILEMAN project.

M. Conti et al. (Eds.): PWC 2003, LNCS 2775, pp. 805–810, 2003.
© IFIP International Federation for Information Processing 2003

knowledge, all these proposals are modified versions of the legacy TCP protocol. However, as shown in Sect. 2, MANETs behave in a completely different way from wired networks, (e.g., Internet), for which the TCP protocol was originally conceived. Therefore, we think that it is more fruitful to think in terms of a new transport protocol optimized for MANETs rather than trying to adapt the TCP protocol to the ad hoc environment. Moreover, the compatibility with host connected to the fixed Internet (i.e., running the TCP protocol) can be achieved by exploiting the Indirect-TCP model [14,11].

In this paper we propose a novel transport protocol, named 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 several innovations with respect to the legacy TCP protocol. In particular, the TPA is able to manage situations that may arise due to nodes' mobility (e.g., route failures and route changes). Furthermore, the congestion control mechanism is completely re-designed with respect to the legacy TCP. Finally, the TPA implements a novel retransmission policy aimed at reducing the number of useless retransmissions and, hence, energy consumption.

2 Motivations

The TCP protocol was originally conceived for wired networks, like the Internet, where nodes are static. However, nodes' movements and failures in MANETs are very frequent, and cause phenomena like *link failures, route failures,* and *route changes*. The TCP is not able to manage such phenomena efficiently. In particular, the sender TCP misinterprets duplicated ACKs and timeouts caused by to route failures or route changes as congestion and activates the congestion control mechanism. This leads to both unnecessary retransmissions and throughput degradation [3,4,10].

Even assuming that nodes in the MANET are static, the MANET behavior is significantly different from that of a traditional wired network. In traditional wired networks, like the Internet, packet losses are almost totally due to congestions causing buffer overflows at intermediate routers. This is not true in MANETs where buffer overflows at intermediate nodes are rare events, while packet losses due to link-layer contentions are largely predominant. The TCP protocol reacts to such packet losses by activating the legacy congestion-control mechanism. A severe drawback of using this mechanism is that the TCP window size is allowed to grow beyond its optimal value, which is typically very small (i.e., 1 - 4 TCP packets [9,10]). This behavior exacerbates the problem, since it produces new link-layer congestions at intermediate nodes.

3 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 TCP protocol and are thus

omitted for the sake of space. In the following we only describe the data transfer phase.

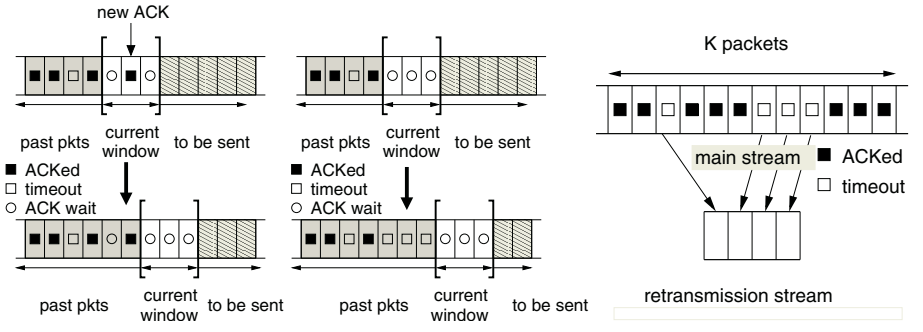
The TPA protocol is based on a sliding-window scheme where the window size varies dynamically according to the flow control and the congestion control algorithms (like the TCP protocol [15]). The congestion control mechanism is described in Sect. 3.3, while the flow control mechanism is similar to the corresponding TCP mechanism [15] and is thus omitted. The 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. Specifically, the source TPA grabs a number of bytes – corresponding to K TPA packets – from the transmit buffer[1], 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 the 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. Please note that it is possible to acknowledge more than one packet by setting the corresponding bits in the bitmap.

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) evaluates the new window size according to the congestion and flow control algorithms; ii) shifts the window forward, so that it starts with the packet next to the last acknowledged one; and iii) sends packets included in the current window (see Fig. 1-a). On the other hand, if all timeouts related to packets in the current window expire, the source TPA still executes steps i)-iii) above, just as in the case the last outstanding packet has been acknowledged (see Fig. 1-b). In other words, the TPA performs a transmission round during which it tries to send all packets within the block, without retransmitting missed packets.

After the first one, the sender performs a second round for retransmitting packets in the block not yet acknowledged, which are said to form a "retransmission stream" (see Fig.1-c). Again, this stream is managed according to steps i)–iii) above. If a packet within the retransmission stream is acknowledged before being retransmitted, it is dropped from the stream. This procedure is repeated until all packets within the original block have been acknowledged by the destination.

The proposed scheme has several advantages with respect to the retransmission scheme used in the TCP. First, the probability of useless retransmissions

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



(a) ACK received (b) all timeouts (c) retr. stream

Fig. 1. Management of the sender sliding window (a,b) and management of the retransmission stream (c)

is reduced since packets for which the ACK is not received before the timeout expiration are not retransmitted immediately (as in the TCP protocol). This is particularly important in MANETs where nodes are highly mobile and, thus, the timeout value might not reflect the current RTT of the connection (see also Sect. 3.2). It should also be observed that the longer waiting time in the TPA protocol does not result in a throughput degradation since during this time interval the sender transmits other packets. Second, the 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 the out-of-order arrivals of packets. This implies that the TPA can operate efficiently also in MANETs using multi-path forwarding [16], where, on the contrary, the TCP performs very poorly [5].

The TPA protocol also includes some mechanisms to dynamically adapt to the network conditions. Specifically, it is able to detect and manage three kinds of events: *route failures*, *route changes* and *congestions*.

3.1 Route Failure Management

Like many other solutions [3,4,5,6], the TPA protocol relies on the network-layer support to detect route failures [1,2]. Specifically, if an intermediate node realizes that a packet cannot be forwarded to the next node because of a link failure, and no alternative route to the destination is available, it sends an *Explicit Link Failure Notification* (ELFN) back to the sender node. At the sender node the ELFN is notified by the network layer to the transport layer. Upon receiving an ELFN, the source TPA enters into a *freeze* state where it refrains from transmitting new packets (there is no available path to the destination).

We assume that the network layer does not provide route re-establishment notifications. Therefore, while in the freeze state, the source TPA sends a probe packet every t_{pr} seconds in order to look for a new route. Upon receiving an ACK

from the receiver it realizes that the route has been re-established. Therefore, it i) leaves the freeze state; ii) sets the congestion window to the maximum value $cwnd_{max}$; and iii) sends the packet that has originated the ELFN.

3.2 Route Change Management

Similarly to the TCP, the TPA protocol estimates the RTT of the connection and, then, uses these estimate to set the retransmission timeout. Both parameters are derived in the same way as in the TCP protocol, i.e.,

$$\begin{aligned} \mu_{RTT}(n) &= g \cdot RTT(n) + (1-g) \cdot \mu_{RTT}(n-1) \\ \sigma_{RTT}(n) &= h \cdot |RTT(n) - \mu_{RTT}(n)| + (1-h) \cdot \sigma_{RTT}(n-1) \quad , \\ Timeout(n) &= \mu_{RTT}(n) + 4 \cdot \sigma_{RTT}(n) \end{aligned}$$

where $\mu_{RTT}(n)$ and $\sigma_{RTT}(n)$ are, respectively, the average value and standard deviation of the RTT estimated at the *n-th* step, $RTT(n)$ is the *n-th* RTT sample, $Timeout(n)$ is the retransmission timeout computed at the *n-th* step and, finally, g and h $(0 < g, h < 1)$ are real parameters (see [15] for details).

When a route change occurs, packets typically experience a variation in the RTT and the retransmission timeout might be no longer appropriate for the new path. To avoid possible useless retransmissions the TPA protocol must detect route changes as soon as they occur, and modify the RTT estimation method accordingly. In practice, the TPA detects that a route change has occurred either i) when a new route becomes available after an ELFN; or ii) when th_{RC} consecutive samples of the RTT are found to be external to the interval $[\mu_{RTT} - \sigma_{RTT}, \mu_{RTT} + \sigma_{RTT}]$. Upon detecting a route change, the TPA replaces the g and h values in the μ_{RTT} and σ_{RTT} estimators to greater values g_1 and h_1 so that the new RTT estimates is heavily influenced by the new RTT sample. 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 values g and h.

3.3 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. 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 is received by the destination TPA, while the source TPA experiences consecutive timeouts without receiving any ELFN. As soon as the source TPA detects th_{CONG} consecutive timeout expirations it assumes that an *ACK inhibition* has occurred, and reacts by entering the *congested* state. This state will be exited when the source TPA receives th_{ACK} consecutive ACKs from the destination.

The TPA congestion control mechanism is window-based as in the TCP protocol. However, as anticipated, in the TPA the maximum congestion window size $cwnd_{max}$ is very small (in the order of 3 or 4 TPA packets). Therefore, the TPA congestion control algorithm is very simple. When the TPA is not in the congested state, the congestion window is set to the maximum value, $cwnd_{max}$. On the other hand, during the congested state, the congestion window is reduced to 1 to allow congestion to disappear.

References

1. J. Broch, D.B. Johnson and D.A. Maltz, "The dynamic source routing protocol for mobile ad hoc networks", Internet Draft of the IETF MANET Working Group, December 1998.
2. C. Perkins, E. Royer and S. Das, "Ad-hoc on demand distance vector (aodv) routing", in IETF Internet Draft, November 2000.
3. 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.
4. G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks", Wireless Networks, Vol. 8, pp. 275–288, 2002.
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. D. Kim, C. Toh and Y. Choi, "TCP-Bus: Improving TCP Performance in Wireless Ad-Hoc Networks", ICC, 2000.
8. F. Wang and Y. Zhang, "Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response", MobiHoc 2002.
9. 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 (CA), March 30–April 3, 2003.
10. S. Xu, T. Saadawi, "Revealing the problems with 802.11 medium access control protocol in multi-hop wireless ad hoc networks", Computer Networks 38 (2002), pp. 531–548.
11. K. Xu, S. Bae, S. Lee and M.Gerla, "TCP Behavior accross Multihop Wireless Networks and the Wired Internet", The Fifth International Workshop on Wireless Mobile Multimedia (WoWMoM 2002), Atlanta (GA), September 28, 2002.
12. P. Gupta and P.R. Kumar, "The Capacity of Wireless Networks", IEEE Transactions on Information Theory, Vol. 46, No. 2, pp. 388–404, March 2000.
13. P. Gupta, R. Gray, and P.R. Kumar, "An Experimental Scaling Law for Ad Hoc Networks", http://black.csl.uiuc.edu/~prkumar/postscript_files.html, 2001.
14. A.Bakre, B.R.Badrinath, "Implementation and Performance Evaluation of Indirect TCP", IEEE Transactions on Computers, Vol.46, No.3, March 1997.
15. W.R. Stevens, "TCP/IP Illustrated", Vol. 1, Addison Wesley, 1994.
16. 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.