Modeling and Analysis of the Trickles Protocol

Mihail Nikitinskiy, Dmitry Chalyy
Demidov Yaroslavl State University
Yaroslavl, Russia
nikitinskiy87@mail.ru, chaly@uniyar.ac.ru

Abstract

Mobile computers and their wireless communications links will be an integral part of the future networks. In the paper we describe an approach for modeling the Trickles protocol which is a novel transport layer protocol. The absence of distributed connection state is a key feature of Trickles which can have significant benefits for mobile networks, for instance to cope with disconnections when mobile host moves from one cell to another.

INDEX TERMS: TRICKLES, TCP, NS-2.

I. INTRODUCTION

Communication protocols, more specifically transport layer protocols, play a key role in data transfer network. The basic purpose of a transport layer protocol is to provide a reliable service for applications information interchange using unreliable network. Today, transport protocols manage about 90% of all traffic in the Internet. Hence, transport protocols are an important topic of today’s research. The main aims of such research projects are improving performance, reliability and correctness of these protocols. Advances in development of open source operating system make possible to implement protocols.

II. TRANSPORT PROTOCOLS

TCP [1] is a major transport protocol of TCP/IP stack. TCP manages information flow using end-to-end connections. TCP has a congestion control mechanism which has been designed to ensure Internet stability along with fair allocation of the network bandwidth.

Let us consider data transfer between web-server and web-browser. Web-browser opens transport connections to web-server and downloads media which have to be rendered. So the state of such a connection is distributed across both sides – server and client. This approach can lead to the following issues:

– misbehaving web-client can lost the state of transport connection due to a failure, so a half-open connection is left which is very difficult to recover;
– dedicating resources at web-server to keeping state invites denial-of-service (DoS) attacks that use up these resources;
– memory is a limiting resource to maintain connections.

To overcome these problems Trickles protocol was proposed in [2]. It is a transport layer protocol which does not maintain distributed connection state. Trickles store data about end-to-end connection on a client side. The client sends requests for a new data to the server. Such a request carries a part of connection state, so the server can realize which data have to be transferred to the client. Server manages network congestion, it can influence on connection state by changing connection information in data packets.

The Trickles protocol simulates the behavior of the TCP by shipping server state to the client side in a transport continuation. The client ships the transport continuation back to the server in each packet, enabling the server to regenerate the state. The sequence of related packets that trace a series of state transitions forms a trickle.

Congestion control algorithm is an important part of the transport protocol which affect network performance. Trickles congestion control algorithm is built upon ideas of TCP Reno [3]. So, it operates in two phases – slow start and congestion avoidance. During slow start server increases number of trickles exponentially. This is made by splitting each incoming trickle (see Fig. 1). Such a behavior makes it possible to quickly reserve available network resources. Congestion avoidance is used for probing available network bandwidth. It is more conservative than slow start, so the number of trickles grows linearly by one per round-trip-time (see Fig. 2).

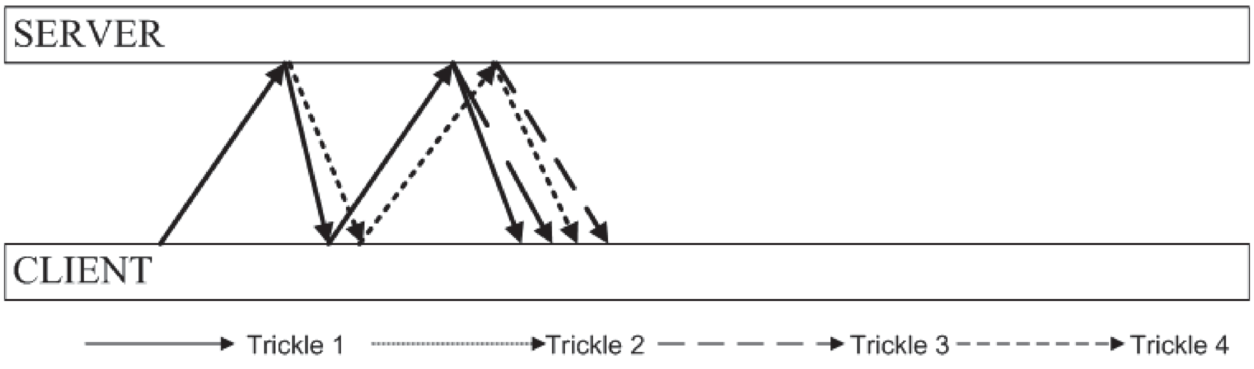

Fig. 1. Exponentianl increase number of trickles during slow start

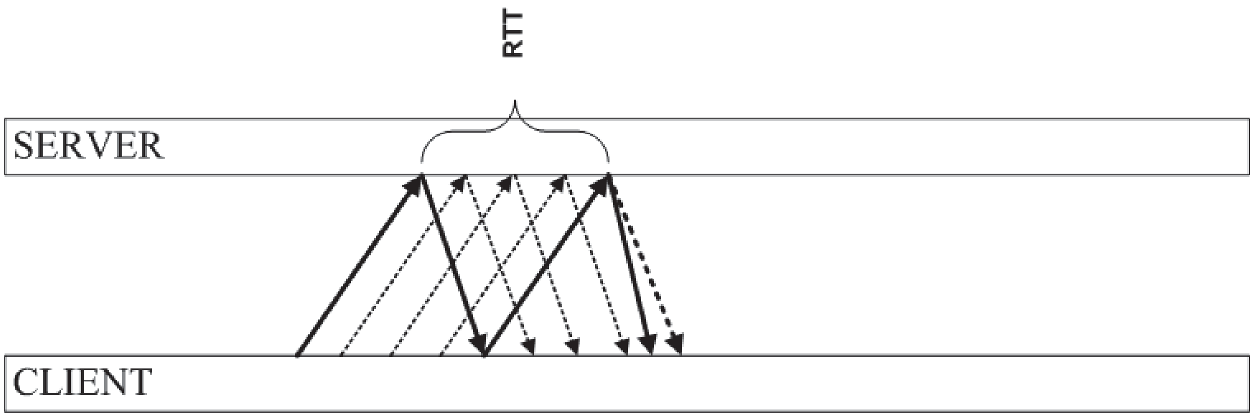

Fig. 2. Linear increase number of trickles during congestion avoidance

Packet losses can occur in the network, so the transport protocol must recover lost data. This is done using retransmissions of packets considered to be lost. There are two kinds of retransmissions. First one is fast recovery. Fast recovery starts when three out-of-order packets are received by the client. Client transmits loss information to the server using SACK

[4]. Server makes necessary retransmissions and halves number of trickles. The last recovery mechanism is retransmission after a timeout.

III. Transport Protocol Modeling

We have built a model of Trickles protocol using ns-2 [5]. NS-2 is a simulation tool which has built-in models of various network protocols. NS-2 proved itself as a suitable tool for performance evaluation of transport protocols (see [6,7,8]).

The essential concept of ns-2 is an agent. Agents represent endpoints where packets are constructed or consumed and are used in the implementation of protocols at various layers. Our model of Trickles is represented by two agents: server agent and client agent. Both agents are implemented as C++ classes which are inherited from the reference Agent class. NS-2 is written in C++ and uses OTcl interpreter as a frontend.

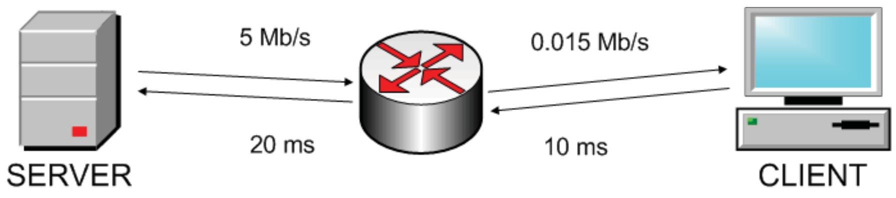

Fig. 3. Test network topology

NS-2 uses OTcl to create network topology which user wants to investigate. For example, we tried our model on the network topology from Fig. 3 using the following script:

```
set ns [new Simulator]
set f [open out.tr w]
$ns trace-all $f
set nf [open out.nam w]
$ns namtrace-all $nf
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
$ns duplex-link $n0 $n1 5Mb 20ms DropTail
$ns duplex-link $n1 $n2 0.015Mb 10ms DropTail
set tserv [new Agent/TricklesServer]
$tserv set packetSize_ 1000
$ns attach-agent $n0 $tserv
set tclient [new Agent/TricklesClient]
$ns attach-agent $n2 $tclient
$ns connect $tserv $tclient
$ns at 0.9 "$tserv filesize -1"
$ns at 1.0 "$tclient start"
$ns at 30.0 "finish"
proc finish {} {
        global ns f nf
        $ns flush-trace
        close $f
        close $nf
        exit 0
}
$ns run
```

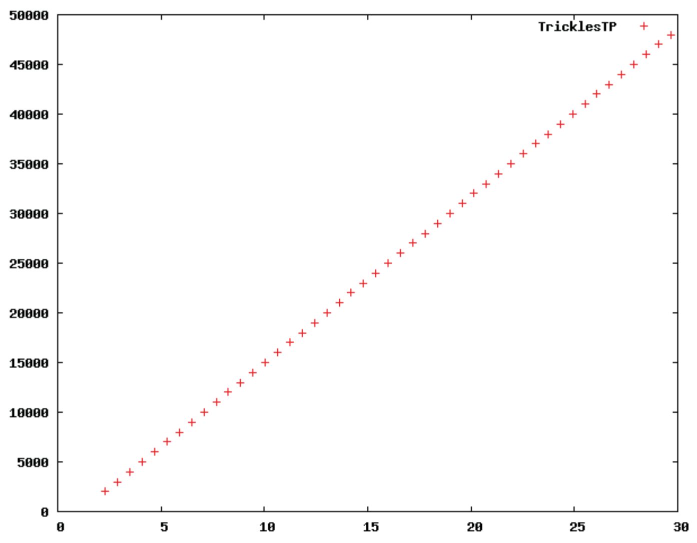

Fig. 4. Data transfer process

The test network topology has three nodes. The node $n0 represents the server, $n1 is an intermediate node and $n2 represents the client. We assume each packet contain 1000 bytes of data. We set that client downloads an infinite amount of data from the server. Transmission starts at 1.0 second of simulator time and finishes at 30.0 seconds. The data transfer process is shown on Fig. 4. We can see that Trickles protocol transmits about 45Kbytes of data for 29.0 seconds on the given network topology.

IV. Conclusion

We described our approach for modeling and performance analysis of the Trickles protocols. Our future work will be devoted to further performance analysis of the protocol. One of the interesting areas is to compare Trickles with different versions of the TCP in various network conditions.

Acknowledgments

Authors would like to thank Professor V.A. Sokolov from Demidov Yroslavl State University for invaluable help and support during the preparation of this work, and also we would like to thank Alan Shieh from Cornell University for his comments on the Trickles algorithm.

References

[1] Transmission Control Protocol. DARPA Internet Program. Protocol Specification. RFC793, September, 1981. Web site: www.rfc-editor.org

[2] A. Shieh, A. C. Myers, and E.G. Sirer “A Stateless Approach to Connection-Oriented Protocols” // ACM Trans. Comput. Syst. Vol. 26, No 3. Article 8. Sept., 2008. 50 p.

[3] M. Allman and V. Paxson and W. Stevens (1999): TCP Congestion Control. RFC 2581 (Proposed Standard). Updated by RFC 3390. Web site: www.rfc-editor.org

[4] M. Mathis, J. Mahdavi , S.Floyd and A.Romanow. TCP Selective Acknowledgment Options. RFC2018, October, 1996. Web site: www.rfc-editor.org

[5] K. Fall, K. Varadhan. The ns Manual (formerly ns Notes and Documentation). January 6, 2009.

[6] G. Holland, N. Vaidya "Analysis of TCP Performance over Mobile Ad Hoc Networks" // Wireless Networks. Vol. 8, 2002. p. 275-288.

[7] M. Bateman, S. Bhatti, G. Bigwood, D. Rehunathan, C. Allison, T. Henderson, D. Miras "A Comparison of TCP Behaviour at High Speeds Using ns-2 and Linux" // Proc. of the 11th Communications and Networking Symposium, Ottawa, Canada, p. 30-37.

[8] L.A. Grieco, S. Mascolo "Performance Evaluation and Comparison of Westwood+, New Reno and Vegas TCP Congestion Control" // ACM SIGCOMM Computer Communications Review. Vol. 34, Issue 2. 2004. p. 25-38.