Protocol Enhancements for Disruption Tolerant Mobile Networking

Simon Schütz, Matus Harvan, Lars Eggert, Stefan Schmid and Marcus Brunner

Abstract — **Internet users are increasingly mobile. Their hosts are often only intermittently connected to the Internet, due to the use of different access networks, gaps in wireless coverage or explicit user choice. When such hosts communicate using the standard Internet protocols, intermittent connectivity can significantly decrease performance and even cause connections to fail altogether. This demonstration visualizes the benefits of protocol enhancements that make Internet communication more tolerant to network disruptions. These enhancements significantly increase performance and avoid connection aborts in the presence of intermittent connectivity.**

I. Introduction

WITHIN the last decade, nomadic networking has become common. Users frequently connect to the Internet with mobile devices such as notebooks, handhelds or mobile phones, using a wide variety of wireless and wired access technologies, including cellular networks (GSM, UMTS), wired or wireless local area networks, or other broadband connections (DSL, cable networks.) In between connected periods, these devices frequently experience disconnected periods – either voluntarily, *e.g.*, based on user preference, or involuntary, *e.g.*, due to bad coverage. This *intermittent connectivity* is a key characteristic of nomadic networking.

Many Internet protocols, including the Internet Protocol (IP) [5] and the Transmission Control Protocol (TCP) [6], do not operate well under intermittent connectivity and in the presence of host mobility. When they were designed, most hosts were stationary and interconnected through permanent, wired links. Support for highly mobile hosts that frequently switch access technologies was not a design objective.

The current Internet protocols have three main shortcomings in intermittently connected scenarios. These shortcomings can significantly decrease the performance of standard TCP connections across intermittently connected paths and can even lead to complete connection failures. They are address changes, disconnection duration and retransmission behavior.

This demonstration showcases TCP enhancements that mitigate all three limitations and enable efficient operation across intermittently connected paths. The improved TCP does not abort connections during disconnected periods and efficiently transmits data during any connected periods. The proposed solution combines two new TCP extensions, the TCP User Timeout Option [3] and the TCP Immediate Retransmission [2], with the Host Identity Protocol (HIP) [7]. See Section III for details of the solution.

Manuscript received February 23, 2006. Simon Schütz, Matus Harvan, Lars Eggert, Stefan Schmid and Marcus Brunner are with NEC Europe Ltd, Network Laboratories, Heidelberg, Germany (e-mail: {schuetz | eggert | schmid | brunner}@netlab.nec.de, m.harvan@iu-bremen.de).

II. Demonstration Description

Figure 1 illustrates the demonstration scenario, in which a mobile user who travels by train is streaming live video to a host in the Internet. The mobile user connects to the Internet through several different wireless base stations that are set up along the train track. The base stations do not fully cover the route of the train track; the user therefore experiences connected periods that alternate with disconnected periods.

To visualize the performance benefits of the TCP enhancements, the mobile host simultaneously streams the same live video to a peer host *twice*, once using standard TCP and once using the enhanced TCP. The peer host displays the two received video streams side-by-side, allowing a direct, visual comparison of the performance differences. The enhanced TCP connection is protected from aborts caused by prolonged connectivity disruptions and the video stream it carries restarts more quickly after a connectivity disruption, compared to standard TCP. The enhanced TCP is also able to utilize short periods of connectivity for transmissions, whereas standard TCP may fail to transmit data during such short connectivity periods.

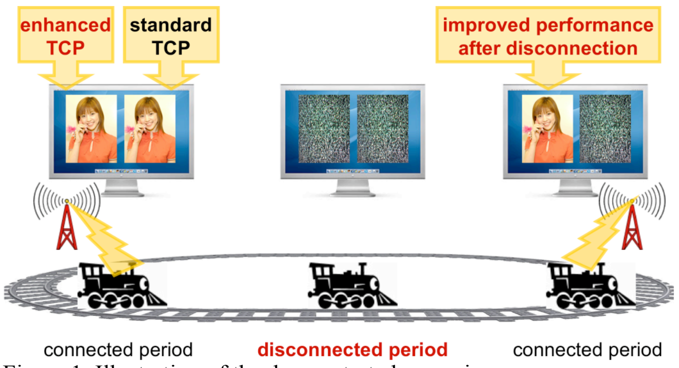

Figure 1. Illustration of the demonstrated scenario.

The demonstration realizes this scenario through use of a *Lego* train that carries a camera-equipped laptop around a train track, shown in the photograph in Figure 2. A stationary PC provides two different wireless access networks to the mobile user; only one is active at a given time and the PC periodically disrupts connectivity when disabling one access network before enabling the second one.

Please note that TCP is obviously not an ideal transport protocol for live video in a production system. For this demonstration, however, using TCP to carry video enables direct, visual comparison of the performance increases, allowing the spectators to develop an intuitive feel for the achievable improvements.

Figure 2. The *Lego* train and track, and the mobile, camera-equipped host used in the demonstration.

III. Technical Details

This section describes the details of the demonstrated TCP enhancements for disruption tolerance. The improved protocol closely approximates the ideal transmission behavior in the presence of intermittent connectivity illustrated in Figure 3. The enhanced TCP utilizes available periods of connectivity almost fully for transmission (from t_0 to t_1 and after t_2) and does not abort the connection during the disconnected period between t_1 and t_2. The proposed solution combines two new TCP extensions, the TCP User Timeout Option [3] and the TCP Retransmission Trigger [2], with the Host Identity Protocol (HIP) [7].

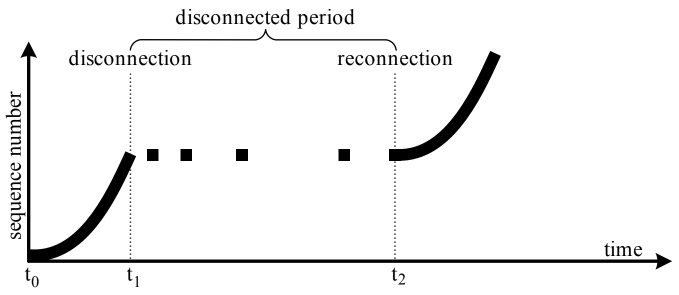

Figure 3. Illustration of enhanced TCP behavior under intermittent connectivity.

A. Disconnection Duration

TCP defines a *user timeout* that specifies the maximum time that data may remain unacknowledged by the peer. The intent behind the *user timeout* is to periodically reclaim system resources allocated to stale connections, where the peer has gone away. Typically, TCP implementations use a system-wide user timeout of a few minutes [10]. Peers do not need to use the same user timeout. After one peer has aborted its half of a connection, a communication attempt by the other peer will result in a reset, as shown in Figure 4. Note that the user timeout is different from the socket timeout provided by socket API. The socket timeout defines how long a send or receive call to the socket waits until data can be written to the send buffer or read from the receive buffer, whereas the user timeout applies to data already sent to the receiver but remains in the output buffer until it is acknowledged.

During a disconnection, no acknowledgments from peers can reach a node. To TCP, this appears as if the connections have gone stale. It will consequently abort them to reclaim associated system resources when the user timeout expires. Connection aborts are error conditions for applications and can cause undesirable effects.

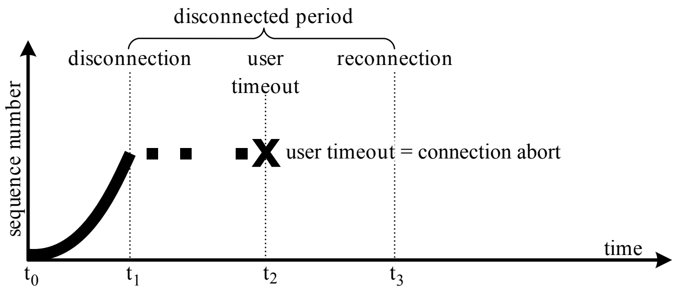

Figure 4. Illustration of standard TCP timeout behavior under intermittent connectivity.

The TCP User Timeout Option [3] allows conforming hosts to exchange per-connection abort timeout requests. This allows mobile hosts to maintain TCP connections across disconnected periods that are longer than their system's default user timeout. A second use of the TCP User Timeout Option is exchange of shorter-than-default user timeouts. This can allow busy servers to explicitly notify their clients that they will maintain the state associated with established connections only across short periods of disconnection.

TCP User Timeout Options allow hosts to both request specific user timeouts for new connections and to request changes to the effective user timeouts of established connections. The latter allows connections to start with short timeouts and only request longer timeouts when external information suggests that disconnection was imminent, and only for connections considered important. The ability to request changes to user timeouts of established connections is also useful to raise the user timeout after in-band authentication has occurred. For example, peers could request longer user timeouts for the TCP connections underlying two-way authenticated TLS connections after their authentication handshakes.

B. Retransmission Behavior

When a disconnection occurs along the path between a host and its peer while the host is transmitting data, it stops to receive acknowledgments. After the *retransmission timeout* (RTO) expires, the host attempts to retransmit the first unacknowledged segment. TCP's recommended RTO management procedure [11] doubles the RTO after each failed retransmission attempt until the RTO exceeds 60 seconds.

This retransmission behavior, shown in Figure 5, is inefficient in the presence of intermittent connectivity. When a disconnection ends, many TCP implementations still wait until the RTO expires before attempting retransmission. Depending on when connectivity becomes available again relative to the next scheduled RTO, this behavior can waste up to a minute of connection time for TCP implementations that follow the recommended RTO management and even more for others.

The TCP Immediate Retransmission [2], the second component of the enhanced TCP, augments the standard retransmission scheme. It uses connectivity indicators, which trigger immediate retransmissions and depend on the specifics of a node and its environment, such as the link-layer technologies it attaches to or the presence of network-layer mechanisms such as DHCP, MobileIP or HIP. The IETF's Detection of

Network Attachment (DNA) working group currently investigates the specifics of providing such indicators (triggers).

Connectivity indicators are generally *asymmetric*, *i.e.*, one may occur on one peer host but not the other. A local event at one host may signal the retransmission trigger, while the other host is unable to detect this event across the network. *Symmetric* connectivity indicators are a special case and always occur concurrently at both communicating hosts. They are usually based on handshake events such as IKE exchanges or HIP readdressing. Symmetric connectivity indicators are an important special case, because the TCP retransmission procedure required in response to a symmetric connectivity indicator is simpler than that for an asymmetric one. The next section will describe this in detail.

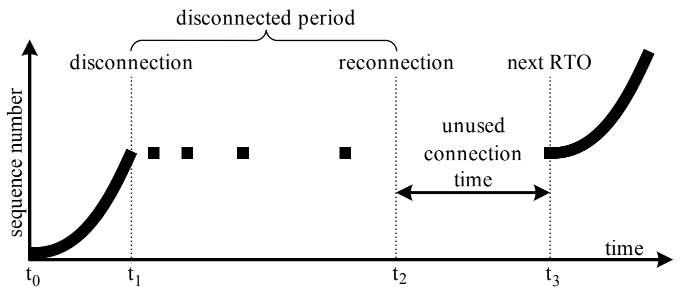

Figure 5. Illustration of standard TCP retransmission behavior under intermittent connectivity

When receiving a symmetric or asymmetric connectivity indicator, conforming TCP implementations immediately initiate the standard retransmission procedure, as if the RTO had just expired. If the connectivity indicator is symmetric, *i.e.*, both peers receive it simultaneously; this simple change is sufficient to kick-start a TCP connection.

If the connectivity indicator is asymmetric, a simple retransmission by one peer is not sufficient. Asymmetric connectivity indicators only occur at one peer but not the other. If the host receiving the trigger has no (or too little) unacknowledged data awaiting retransmission, it will not emit enough segments to cause the peer node, which may have unacknowledged data, to attempt a retransmission itself. The retransmission trigger would thus only function in one direction, which is ineffective for asymmetric communication. To avoid this issue, conforming TCP implementation thus perform a different retransmission procedure in response to an asymmetric connectivity indicator.

C. Mobility Management

Finally, switching between access networks – due to physical mobility or not – usually also changes a host's topological location in the network. Because it is no longer reachable at its old address, a host must start to use a different IP address that identifies its new network attachment point.

Many Internet transport protocols, including TCP, use IP addresses (often together with other identifiers such as port and protocol numbers) to uniquely and permanently identify the endpoints of transport-layer connections during their lifetime. When a host changes its IP address, the local endpoints of its established TCP connections thus change as well. Whereas some newer transport protocols such as SCTP [8] support changing the endpoints of established connections and can continue to operate across IP address changes, TCP does not. Consequently, TCP implementations must abort established connections when IP addresses change. This error condition, illustrated in Figure 6, is visible to applications and may cause them in turn to abort or behave undesirably.

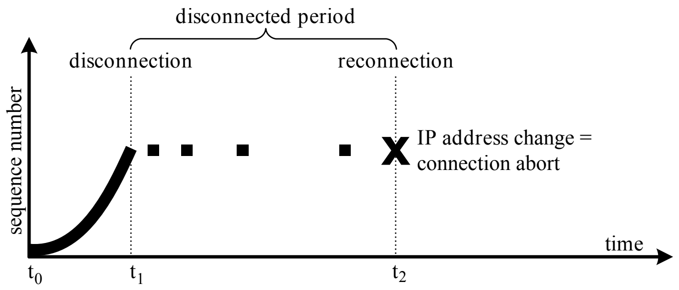

Figure 6. Illustration of TCP behavior in the presence of IP address changes.

The Host Identity Protocol (HIP) [7] and its mobility and multi-homing extensions [9] provide the necessary mobility support for hosts that roam across different access networks. The HIP layer, a new shim layer between the network and transport layers, enables host mobility that is transparent to applications and services. Mobility support for established connections is provided end-to-end. Each host informs its current peers using a three-way handshake when IP address changes occur. Note that this three-way handshake can serve as a symmetric connectivity indicator for the TCP Immediate Retransmission.

ACKNOWLEDGMENT

This demo is partly a product of *Ambient Networks*, a research project supported by the European Commission under its *Sixth Framework Program*. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the *Ambient Networks* project or the European Commission.

REFERENCES

[1] Simon Schütz, Lars Eggert, Stefan Schmid and Marcus Brunner. Protocol Enhancements for Intermittently Connected Hosts. *ACM Computer Communication Review (CCR)*, Vol. 35, No. 3, July 2005, pp. 5-18.
[2] Lars Eggert, Simon Schütz and Stefan Schmid. TCP Extensions for Immediate Retransmissions. *Internet Draft* draft-eggert-tcpm-tcp-retransmit-now-02, Work in Progress, June 2005.
[3] Lars Eggert and Fernando Gont. TCP User Timeout Option. *Internet Draft* draft-ietf-tcpm-tcp-uto-02, Work in Progress, October 2005.
[4] Simon Schütz. Network Support for Intermittently Connected Mobile Nodes. *Diploma Thesis*, University of Mannheim, Germany, June 2004.
[5] Jon Postel. Internet Protocol. *STD 5, RFC 791*, September 1981.
[6] Jon Postel. Transmission Control Protocol. *STD 7, RFC 793*, September 1981.
[7] Robert Moskowitz and Pekka Nikander. Host Identity Protocol Architecture. *Internet Draft* draft-ietf-hip-arch-03, Work in Progress, August 2005.
[8] Randall R. Stewart, Qiaobing Xie, Ken Morneault, Chip Sharp, Hanns Juergen Schwarzbauer, Tom Taylor, Ian Rytina, Malleswar Kalla, Lixia Zhang and Vern Paxson. Stream Control Transmission Protocol, *RFC 2960*, October 2000.
[9] Pekka Nikander, Jari Arkko and Tom Henderson. End-Host Mobility and Multi-Homing with Host Identity Protocol. *Internet Draft* draft-ietf-hip-mm-01, Work in Progress, February 2005.
[10] W. Richard Stevens. TCP/IP Illustrated, Volume 1: The Protocols. Addison-Wesley, 1994.
[11] Vern Paxson and Mark Allman. Computing TCP's Retransmission Timer. *RFC 2988*, November 2000.