Mapping the BEEP Core onto TCP

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

   This memo describes how a BEEP (Blocks Extensible Exchange Protocol)
   session is mapped onto a single TCP (Transmission Control Protocol)
   connection.

Table of Contents

1. Introduction

   This memo describes how a BEEP [1] session is mapped onto a single
   TCP [2] connection. Refer to Section 2.5 of [1] for an explanation
   of the mapping requirements.

2. Session Management

   The mapping of BEEP session management onto the TCP service is
   straight-forward.

   A BEEP session is established when a TCP connection is established
   between two BEEP peers:

   o   the BEEP peer that issues a passive TCP OPEN call is termed the
       listener; and,

   o   the BEEP peer that issues an active TCP OPEN call is termed the
       initiator.

   A simultaneous TCP OPEN would result in both BEEP peers believing
   they are the initiator and neither peer will be able to start any
   channels. Because of this, services based on BEEP must be designed
   so that simultaneous TCP OPENs cannot occur.

   If both peers agree to release a BEEP session (c.f., [1]'s Section
   2.4), the peer sending the "ok" reply, immediately issues the TCP
   CLOSE call. Upon receiving the reply, the other peer immediately
   issues the TCP CLOSE call.

   A BEEP session is terminated when either peer issues the TCP ABORT
   call, and the TCP connection is subsequently aborted.

3. Message Exchange

   The mapping of BEEP exchanges onto the TCP service is less straight-
   forward.

   Messages are reliably sent and received using TCP's SEND and RECEIVE
   calls. (This also provides ordered delivery of messages on the same
   channel.)

   Although TCP imposes flow control on a per-connection basis, if
   multiple channels are simultaneously in use on a BEEP session, BEEP
   must provide a mechanism to avoid starvation and deadlock. To
   achieve this, BEEP re-introduces a mechanism used by the TCP:
   window-based flow control -- each channel has a sliding window that
   indicates the number of payload octets that a peer may transmit
   before receiving further permission.

3.1 Flow Control

   Recall from Section 2.2.1.2 of [1] that every payload octet sent in
   each direction on a channel has an associated sequence number.
   Numbering of payload octets within a data frame is such that the
   first payload octet is the lowest numbered, and the following payload
   octets are numbered consecutively.

   The actual sequence number space is finite, though very large,
   ranging from 0..4294967295 (2**32 - 1). Since the space is finite,
   all arithmetic dealing with sequence numbers is performed modulo
   2**32. This unsigned arithmetic preserves the relationship of
   sequence numbers as they cycle from 2**32 - 1 to 0 again. Consult
   Sections 2 through 5 of [3] for a discussion of the arithmetic
   properties of sequence numbers.

3.1.1 Channel Creation

   When a channel is created, the sequence number associated with the
   first payload octet of the first data frame is 0, and the initial
   window size for that channel is 4096 octets. After channel creation,
   a BEEP peer may update the window size by sending a SEQ frame
   (Section 3.1.3).

   If a BEEP peer is asked to create a channel and it is unable to
   allocate at least 4096 octets for that channel, it must decline
   creation of the channel, as specified in Section 2.3.1.2 of [1].
   Similarly, during establishment of the BEEP session, if the BEEP peer
   acting in the listening role is unable to allocate at least 4096
   octets for channel 0, then it must return a negative reply, as
   specified in Section 2.4 of [1], instead of a greeting.

3.1.2 Sending Messages

   Before a message is sent, the sending BEEP peer must ensure that the
   size of the payload is within the window advertised by the receiving
   BEEP peer. If not, it has three choices:

   o   if the window would allow for at least one payload octet to be
       sent, the BEEP peer may segment the message and start by sending a
       smaller data frame (up to the size of the remaining window);

   o   the BEEP peer may delay sending the message until the window
       becomes larger; or,

   o   the BEEP peer may signal to its application that it is unable to
       send the message, allowing the application to try again at a later
       time (or perhaps signaling its application when a larger window is
       available).

   The choice is implementation-dependent, although it is recommended
   that the application using BEEP be given a mechanism for influencing
   the decision.

3.1.3 Processing SEQ Frames

   As an application accepts responsibility for incoming data frames,
   its BEEP peer should send SEQ frames to advertise a new window.

   The ABNF [4] for a SEQ frame is:

       seq   = "SEQ" SP channel SP ackno SP window CR LF

       ackno   = seqno

       window   = size

       ; channel, seqno, and size are defined in Section 2.2.1 of [1].

   The SEQ frame has three parameters:

   o   a channel number;

   o   an acknowledgement number, that indicates the value of the next
       sequence number that the sender is expecting to receive on this
       channel; and,

   o   a window size, that indicates the number of payload octets
       beginning with the one indicated by the acknowledgement number
       that the sender is expecting to receive on this channel.

   A single space character (decimal code 32, " ") separates each
   component. The SEQ frame is terminated with a CRLF pair.

   When a SEQ frame is received, if any of the channel number,
   acknowledgement number, or window size cannot be determined or is
   invalid, then the BEEP session is terminated without generating a
   response, and it is recommended that a diagnostic entry be logged.

3.1.4 Use of Flow Control

   The key to successful use of flow control within BEEP is to balance
   performance and fairness:

o large messages should be segmented into frames no larger than
   two-thirds of TCP's negotiated maximum segment size;

o frames for different channels with traffic ready to send should be
   sent in a round-robin fashion;

o each time a frame is received, a SEQ frame should be sent whenever
   the window size that will be sent is at least one half of the
   buffer space available to this channel; and,

o if the transport service presents multiple frames to a BEEP peer
   simultaneously, then a single consolidating SEQ frame may be sent.

In order to avoid pathological interactions with the transport
service, it is important that a BEEP peer advertise windows based on
available buffer space, to allow data to be read from the transport
service as soon as available. Further, SEQ frames for a channel must
have higher priority than messages for that channel.

Implementations may wish to provide queue management facilities to
the application using BEEP, e.g., channel priorities, (relative)
buffer allocations, and so on. In particular, implementations should
not allow a given channel to monopolize the underlying transport
window (e.g., slow readers should get small windows).

In addition, where possible, implementations should support transport
layer APIs that convey congestion information. These APIs allow an
implementation to determine its share of the available bandwidth, and
also be notified of changes in the estimated path bandwidth. Note
that when a BEEP session has multiple channels that are
simultaneously exchanging large messages, implementations without
access to this information may have uncertain fairness and progress
properties during times of network congestion.

Finally, implementors should follow the guidelines given in the
relevant portions of RFC1122 [5] that deal with flow control (and
bear in mind that issues such as retransmission, while they interact
with flow control in TCP, are not applicable to this memo). For
example, Section 4.2.2.16 of RFC1122 [5] indicates that a "receiver
SHOULD NOT shrink the window, i.e., move the right window edge to the
left" and then discusses the impact of this rule on unacknowledged
data. In the context of mapping BEEP onto a single TCP connection,
only the portions concerning flow control should be implemented.

4. Security Considerations

Consult Section [1]'s Section 9 for a discussion of security issues.

References

[1]   Rose, M., "The Blocks Extensible Exchange Protocol Core", RFC
      3080, March 2001.

[2]   Postel, J., "Transmission Control Protocol", STD 7, RFC 793,
      September 1981.

[3]   Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982,
      August 1996.

[4]   Crocker, D. and P. Overell, "Augmented BNF for Syntax
      Specifications: ABNF", RFC 2234, November 1997.

[5]   Braden, R., "Requirements for Internet Hosts -- Communication
      Layers", STD 3, RFC 1122, October 1989.

Author's Address

Marshall T. Rose
Invisible Worlds, Inc.
1179 North McDowell Boulevard
Petaluma, CA 94954-6559
US

Phone: +1 707 789 3700
EMail: mrose@invisible.net
URI: http://invisible.net/

Appendix A. Acknowledgements

   The author gratefully acknowledges the contributions of: Dave
   Crocker, Steve Harris, Eliot Lear, Keith McCloghrie, Craig Partridge,
   Vernon Schryver, and, Joe Touch. In particular, Dave Crocker
   provided helpful suggestions on the nature of flow control in the
   mapping.