Request for Comments: NP
Obsoletes: PPG
Category: Standards Track
03 May 2001

Bruce Molay
Harvard University

Simple Ping Pong Ball Transfer Protocol Version 1

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.

Table of Contents

1. Introduction

The pong game described in *Understanding Unix/Linux Programming* allows one player to bounce the letter 'o' (or, optionally, the letter 'O', the digit zero, or any of other available text characters) around a three-sided court on a text-based terminal display.

This memo describes a communication system that will allow two players to pass the letter 'o' (or optionally other characters, see above) from one instance of the pong game to another.

The purpose of this communication system is to allow programs to simulate a ping pong game. The player who starts the first game is called the server game, and the player that attaches to that server game is called the client game.

2. A Short Digression

This protocol describes the communication between a server game and a client game. There need not be human players at either end. The protocol does not specify the properties of the court on either end of the communication, nor does it specify the paddle size or action.

There is nothing in this memo that would prevent either game from being a program that simply generates PPB's (the serving machine) or a program that consumes PPB's (a PPB sink).

3. Basic Operation

Initially, the server game starts the SPPBTP service by listening on a port of its choice. when a client game wishes to play a game with the server, it establishes a TCP connection with the server game. When the connection is established, the SPPBTP server sends a 'hello' greeting and the client sends a 'my name is' greeting. The client and server then exchange commands until the game is over, one player quits, or the connection is closed or aborted.

Commands in the SPPBTP consist of a case-insensitive keyword, possibly followed by one or more arguments. All commands are terminated by a CRLF pair. Keywords and arguments consist of printable ASCII characters. Keywords and arguments are each separated by a single SPACE character. Keywords are four characters long. Each argument may be up to 40 characters long. The total length of a line may not exceed 128 characters including the CRLF pair at the end.

A SPPBTP session begins in the INTRODUCTION state and then, once the first PPB is put in play, the session moves to the PLAYBALL state. The session remains in the PLAYBALL state until a player loses, a player quits, or the TCP connection is dropped.

Each player MUST respond to an unrecognized, unimplemented, or syntactically invalid command by responding with a negative status indicator. A player MUST respond to a command issued when the session is in an incorrect state by responding with a negative status indicator.

A negative status indicator is the string ?ERR and may be followed by a message of explanatory text.

Each SPPBTP player MAY have an inactivity autologout timer. Such a timer MUST be of at least 5 minutes' duration. The receipt of any command from the client during that interval should suffice to reset the autologout timer. When the timer expires, the player may treat that event as though it had received a QUIT command from the other game.

4. The INTRODUCTION State

Once the TCP connection has been opened by a SPPBTP client, the two players introduce themselves and discuss the parameters of the game.

First, the SPPBTP server issues a ‘welcome to the game’ command. This message is of the form:

```
S:  HELO version ticks_per_second netheight player_name
```

The following example,

```
S:  HELO 1.0 1000 16 Ann
```

tells the client that the server is speaking SPPBTP version 1.0, that it has a timer set at 1000 ticks per second, that the net is 16 character cells tall, and that the player’s name is Ann.

Second, the client responds with a message saying ‘thanks, my name is..’ This message has the form:

```
C:  NAME version player_name
```

The following example,

```
C:  NAME 1.0 Bob
```

tells the server that the human playing the game is named Bob.

If the client and server report different protocol versions during the introduction, both programs must use the lower of the two reported protocols. If a program is not willing to use the lower protocol, it will send the QUIT message and terminate the game.

Finally, the server sends a message saying ‘since you are the guest, why don’t you serve the first PPB..’ This message has the form:

```
S:  SERV number_of_PPB’s
```

The following sample message,

```
S:  SERV 3
```

tells the client to serve a PPB and to consider the game over when three PPB’s have gone out of bounds.

5. The PLAYBALL State

Once the game begins, the two programs toss the PPB from side to side. The session is said to be in the PLAYBALL state. In this state, only one player has the PPB at a time. Each program, then is in one of two states, the PLAY state or the WAIT state.

When a player has the PPB, that program is said to be in the PLAY state. The PPB moves around its screen subject to the usual pong operation. During that time, the opposing program is in the WAIT state.

When the PPB reaches the net the program with the ball sends the PPB to the other program by sending a BALL command to the other program. The program sending the BALL command now changes to the WAIT state, and the program receiving the BALL command changes from the WAIT state to the PLAY state.

When a player misses the PPB allowing it to go out of bounds, that program sends a MISS message to the other program. The MISS message says to the other player ‘I just missed the PPB, it is your turn to serve.’

If the initial number of PPB’s has been exhausted the game is over. The two programs shake hands and quit.

Here are the SPPBTP commands valid in the PLAYBALL state:

```
BALL net_position xttm yttm ydir [PPBchar]

   Arguments:
      net_position: the offset from the top of the net
      xttm, yttm  : x and y ticks to move settings
      ydir        : 1 or -1 specifying y direction
      PPBchar     : an optional argument for the symbol

   Restrictions:
      May only be received while program is in WAIT state
      May only be sent while program in in PLAY state

   Discussion:
      The program sends a PPB ‘over the net’ with the BALL
      command.  The BALL command tells the receiving game
      about the PPB.  The four required parameters are enough
      information for the receiving side to get the PPB in
      play on its end.

      The net_position is the position of the PPB relative
      to the top of the net.  The xttm and yttm values
      represent the time to move in each direction in ‘ticks’.

      The ydir tells whether the PPB is currently moving in
      the positive y direction or in the negative y direction.
      Positive y means downwards on the screen.

      The optional argument is a single character.  It is a
      suggestion that the other game use that character to
      represent the PPB on its screen.  It may be ignored.
```

Possible Responses:

BALL	: the opponent bounces the PPB back
MISS	: the opponent misses the PPB
QUIT	: the opponent quits the game

Effect:
The game that receives the BALL command enters the PLAY state. The game sending the BALL command enters the WAIT state.

Examples:
BALL 5 200 420 1 @

MISS [msg]

Arguments:
An optional string said by the player who missed the PPB. Values might be things like ‘good shot!’ or ‘drat!’ or ‘That was too fast.’

Restrictions:
May only be sent while in the PLAY state, may only be received while in the WAIT state.

Discussion:
When a player misses the PPB, the program sends a MISS message to the other game. The MISS command may contain a text string as an argument. The receiving program may print out the message to provide a sense of lively repartee between the players.

When a game in the WAIT state receives the MISS message, it will serve a new PPB if there are more PPB’s to play. If there are no more PPB’s to play, the receiver will reply DONE.

Possible Responses:

BALL	: the opponent serves a new PPB
MISS	: the opponent served a PPB and missed it
DONE	: the opponent says the game is done
QUIT	: the opponent quits the game

Example:
MISS The sun was in my eyes

6. Ending the Game

Several different conditions cause the game to end. The last PPB goes out of play, one player may quit, or the connection may be lost. The game is done when the last PPB goes out of play and is reported by the transmission of a MISS command.

When a player receives a MISS command, that player is expected to serve the next PPB. If there are no more PPBs left, that player sends a DONE command. The player expects a QUIT command as an acknowledgement.

QUIT [msg]

Arguments:
An optional string said by the player who is quitting the game. Values might be things like ‘Thanks for a good game’ or ‘I’ll win next time!’

Restrictions:
May be sent at any time.

Discussion:
When either program receives the QUIT message, it prints the optional message to the screen, if there is an optional message, and then closes the connection and shuts down the session.

DONE [message]

Arguments:
An optional string said by the player who receives the MISS command when there are no more PPBs

Restrictions:
May only be sent by a program in response to a MISS command when the number of PPB’s left equals zero.

May only be received by a program that just sent a MISS command when the number of PPB’s left equals zero.

Discussion:
When a game in the WAIT state receives a MISS message, it determines if the number of PPB’s left is zero. If the number of remaining PPB’s is zero, the game sends a DONE message.

Possible Responses:

QUIT	: the opponent acknowledges DONE

7. Scaling and Operational Concerns

The classic pong game [Molay, 1990] uses a text-based display with a vertical wall consisting of 16 'l' characters. For flexibility with larger text windows or other graphics displays, the server sends in its initial message the height of its net. This allows the other game to set its net height to the same size.

The client game is free to play with a larger or smaller net. It simply has to scale the y-position of the ball. For example, if the client has a net 32 units tall, it can take the y-position given in the BALL command and double it. By doing so, the client will ensure that a ball passing through the middle of the net on the server will appear through the middle of the net on the client.

The ticks per second can also be used to scale timing. If the server says it is using 1000 ticks per second, and the client uses 100 ticks per second, the client may adjust the ttm values to match its resolution.

8. SPPBTP Command Summary

```
INTRODUCTION State

  HELO version ticks_per_second netlength name
  NAME version name
  SERV nPPBs

PLAYBALL State

  BALL net_position xttm yttm ydir [PPBchar]
  MISS [message]
  DONE [message]

Any State

  QUIT

Error Code

  ?ERR [message]
```

9. Example SPPBTP Session

```
Client                          Server
-----------------------------   -------------------------------
                            <wait for connection on TCP port>
<open connection>
                      <-----  HELO 1.0 20 16 mrose
NAME 1.0 dewey  ----------------->
                      <-----  SERV 3
BALL 10 10 12 +1 ---------------->
                      <-----  BALL 6 8 20 -1
MISS  --------------------------->
                      <-----  BALL 2 5 8 +1
BALL 5 5 8 -1  ------------------>
                      <-----  MISS
BALL 8 15 8 +1 ------------------>
                      <-----  BALL 12 6 4 -1
BALL 2 6 4 -1  ------------------>
                      <-----  BALL 10 12 8 +1
BALL 7 20 12 +1 ----------------->
                      <-----  MISS
DONE Nice game, I win ----------->
                      <-----  QUIT Let's play again soon
<close connection>                <close connection>
                            <wait for next connection>
```

10. Security Considerations

We are not aware of any security risks posed by SPPBTP.

11. Acknowledgments

This is the first draft of a protocol for transmitting PPB data over a TCP link. The idea of drafting a uniform standard was proposed by Antonio Aranda Eggermont when he noted that a standard protocol would allow any version of netpong to interoperate with any other.

I also wish to thank Peter Miller for his suggestion for the correct way to pronounce ‘SPPBTP’. We hope he can provide a sound file containing the noise for our web site.

In addition, countless, although no more than a dozen, students offered some good ideas (‘sounds cool!’) and helpful criticism (‘I don’t like computer games’) during the early development phase.

12. Author’s Address

Bruce Molay
Harvard University Division of Continuing Education
51 Brattle Street
Cambridge, MA 02138

EMail: molay@fas.harvard.edu

13. Changes from Previous Version

[a] Version PPG includes more specific details about protocol version handling. In the earlier version, client and server exchanged protocol numbers during the initial dialog, but the RFC did not specify how they were to handle different versions of the protocol.