PPP for Asynchronous PAD to Synchronous X.25 access

Status of this Memo

Distribution of this memo is unlimited.

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress".

To view the entire list of current Internet-Drafts, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Abstract

The PPP protocol allows data transfer thru asynchronous or synchronous connections. But the prevalent Public Switched Data Networks (PSDNs) support connections between asynchronous and synchronous protocols. This document defines extensions to the PPP protocol to support asynchronous PAD to synchronous X.25 protocols on PSDNs.

1. Introduction

The X.25 [10] PSDNs consist of a set of Switches and PADs. The multiuser hosts connect to the Synchronous X.25 ports of the switch and the single user PCs generally connect to the Asynchronous PAD ports (Fig. 1).

One of the major requirements of the users is to run TCP/IP based applications between these PCs and the multiuser hosts on the X.25 PSDN. Currently the following Internet protocols are available -


There is no protocol for the above scenario of TCP/IP access between a Asynchronous serial line at one end and a Synchronous X.25 line at the other. This memo proposes one such protocol.
Fig. 1    A PC accessing a X.25 host through a PAD
2. Requirements from the Protocol

A protocol to be defined for such a purpose must meet the following requirements.

1. It must allow transparent TCP/IP access between the PC connected to the PAD Async line and the multiuser host connected to the Sync X.25 Switch port, under arbitrary segmentation of packets by the network.

2. The protocol must be implementable using the existing set of X.25 equipments - Switches and PADs.

3. The protocol must coexist with other protocol stacks running over the underlying X.25 layers, e.g. 3X PAD[2,3,4], SNDCF, PPP framing in X.25, etc.

3. The Protocol

This protocol is broadly based on the mechanisms defined by the author in [1]. Briefly, the protocol works as follows. Async PPP is run over both the PC and the multiuser host (Fig. 2). The TCP/IP layers are made to run over the PPP layer.

Since the PC connects to PAD via the Async serial link and PPP is defined to work over serial links, PPP protocol on PC is made to run directly over this Async link. The proposed protocol defines mechanisms by which initially a X.25 call is made from the PC Async port to the remote host through the PAD. Once the connection is made, PPP is now made to run over this Async port.

The remote host connects to the network through a X.25 port. Since PPP does not work directly over the X.25 layers, the protocol defines an extra layer of software which resides between the PPP and the underlying X.25 layers. This layer gets incoming packets from X.25 stack, breaks them into individual characters and gives these to the PPP layer above to be interpreted by the protocol.

Generally, PAD interprets some control characters (like the PAD escape character). This is avoided by setting Transparent Profile mode over the PAD Async port. This sends all characters uninterpreted. The data is forwarded when the PAD buffer becomes full or a delay of 1 second is received between any 2 received characters.

Thus, the above mechanisms ensure that a protocol packet sent by the TCP/IP layers on one computer is received in the same format on the other, ensuring transparent working of TCP/IP protocol and applications between the two. We shall now describe different phases of the protocol in detail.
Fig. 2 The protocol layers on the PC and the multiuser host
3.1 Call Establishment Phase

The protocol must work over the existing PADs. Whenever a PC makes an outgoing call through a PAD, the PAD invariably puts the PAD X.29 PID (1,0,0,0) in the first 4 bytes of the X.25 Call Request User Data field. When such an incoming Call Request packet is received by the remote host, it invariably invokes the 3X PAD software to handle the call. This software allows the remote login application to run between the PC and the host.

Since the proposed protocol on PC is also making the call through a PAD, the same PID will be received at the remote host causing the 3X PAD to be run instead of the proposed protocol over the remote host.

This problem is solved by following the following mechanism. The PC software makes a call through the PAD with the Fast Select option. This allows extra data to be sent with the call. The first 4 bytes of this extra data are filled with the pattern (0,0,0,1). The remote host receives an incoming Fast Select call. The PID is same as for any other normal PAD call (1,0,0,0). But since it is a Fast Select call, the host is made to invoke the proposed protocol instead of the 3X PAD. Also the host sends a Fast Select call acceptance packet by changing the first 4 bytes to a pattern (0,0,0,2). The PC on receiving this acceptance packet compares the data with (0,0,0,2). The PC clears the call if the data does not match, else it sends a command to PAD to make it work in Transparent Profile and PPP is invoked over the Async port. Note that if the PAD does not display the data in the Call Accept packet, an alternative approach is suggested where the PC accepts the call and then the multiuser host software sends the first X.25 Data packet with the above pattern. The PC software then compares this data and invokes PPP.

Let us see as to how the proposed protocol coexists with the existing set of protocols running over the X.25 stack. Let the multiuser host be running only 3X PAD software over it and not running our proposed protocol. The above steps must ensure that the PPP process on the PC does not incorrectly start a session with the 3X PAD remote login process on the multiuser host.
On receiving an incoming Fast Select call, the remote 3X PAD process can behave in any of the following two ways. If it does not support the Fast Select facility, it may clear the call. Thus no call is established, as desired. Or else if it supports Fast Select facility it sends an acceptance packet. Since the 3X PAD process protocol is not supposed to send any data in the Call Accept User Data field, it may either send this packet without any data or it may simply copy the incoming pattern in it, depending on its implementation.

The comparison of this field (0,0,0,1) with pattern (0,0,0,2), as above, fails. Thus the PC clears the call and no session is established. Thus in no case, incorrect associations can be made due to the proposed protocol.

3.2 Data Transfer Phase

Once the connection is successfully established, the standard PPP and TCP/IP are running over the connection. The data transfer phase of the protocol will ensure that data is received correctly even in case of arbitrary segmentation in the X.25 network.

Since the PPP at the PC end is running in async. mode, the "Octet-stuffed framing" mechanism is used [8] for data transfer. The PPP layer at the remote host (running above X.25) also uses the same method to interpret the data.

The PPP on PC encloses the TCP/IP packets within headers and trailers and transmits the resultant byte stream to the PAD. Let us assume that PAD had to send it as 2 X.25 packets. The packets reach the X.25 stack on the multiuser host which strips the X.25 headers and hands over the individual packets to proposed protocol layer above it.

The proposed protocol layer works under the control of PPP running above it. It receives X.25 packets from the underlying X.25 stack, breaks these into individual bytes and hands these over to the PPP layer running above it. Each time PPP requires a new packet, it asks for individual bytes from this layer. The steps taken by this layer on receiving a request for a byte from PPP are as follow.
1. If the layer does not possess a X.25 data packet, request for one from the underlying X.25 stack.

Initialise a local pointer to the first byte of the packet.

Extract this byte of the packet and give it to the PPP layer above requesting a byte.

2. Else, if it already possesses a X.25 data packet, give the byte in the packet pointed to by the local pointer.

3. Increment a local pointer to point to the next byte of the packet. If the complete packet has been read, discard the packet.

The PPP layer above waits for getting a start flag and keeps on requesting bytes from this layer till end flag is received. This packet is handed over to TCP/IP layers above it. Thus, PPP is oblivious of the fact that its packet has been received as multiple X.25 packets. When the packet is given to the TCP/IP layers above, it is exactly the same as was transmitted by the sending TCP/IP entity.

Thus it is interpreted correctly and networking applications can run successfully across the two systems.

The sending of data from the multiuser host to the PC is also similar. The PPP hands over the individual bytes to the proposed protocol layer below it. The layer works like a PAD works in the Transparent Profile mode, i.e. sends a X.25 packet when its buffer is full or a gap of 1 second is received between any 2 bytes.

3.3 Call Disconnection Phase

When the TCP/IP application on the PC terminates, it sends a management command to PPP asking it to terminate the call. This makes the PPP to pull down the DTR signal on the Async line. This causes the PAD to send a clear packet to the remote, which clears the VC.
4. Conclusion

We have proposed a protocol which allows TCP/IP access between PCs connected to a PAD and multiuser hosts connected to a X.25 Switch. The protocol works under arbitrary segmentation of packets in the X.25 network. It is implementable on existing set of PADs and Switches and co-exists with the existing set of protocol stacks running over X.25 layers.

5. References


[10] "Draft revised Recommendation X.25 - Interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) for terminals operating in the packet mode and connected to Public Data Networks by dedicated circuit", CCITT, 1992.

6. Author's Address

Vimal K. Khanna,
Mindware,
B-60, Okhla Industrial Area Ph-I,
New Delhi - 110 020, India.

Phone: (91 11) 681 52 04
Email: vimal@pcl.stpn.soft.net
7. Expiration date of the document

31 January, 1998