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 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 "id-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 mechanisms for the PPP protocol to run on asynchronous PAD to synchronous X.25 connections 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 -


One needs mechanisms 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 such mechanisms. A comparison with mechanisms defined in [8] has also been made.
Fig. 1 A PC accessing a X.25 host through a PAD
2. Requirements

The mechanisms to be defined for such a purpose must meet the following requirements.

1. These must allow transparent TCP/IP access between the connected PC and the multiuser host under arbitrary segmentation of packets by the network.

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

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

3. Working

The mechanisms are broadly based on the procedure defined by the author in [1]. Briefly, these work 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.

PC connects to PAD via the Async serial link and PPP protocol is run over this. We define 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 made to run over the 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, we define 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.

We shall now describe different phases in detail.
Fig. 2  The protocol layers on the PC and the multiuser host

Khanna  [Page 3]
3.1 Call Establishment Phase

These mechanisms 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 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 solution 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 PPP over the remote host.

We define the use of first 4 bytes of PAD PID and one more byte of CUD to define the identifier as (1,0,0,0,0xCF). This makes the remote host invoke PPP instead of a login process. The PC on receiving the call acceptance sends a command to PAD to make it work in Transparent Profile and PPP is invoked over the Async port.

Currently, mechanisms defined in [8] require the remote host to await the PPP LCP frames before deciding to invoke PPP or continue with the remote login process. But, since these mechanisms depend on data patterns received after the call establishment, these do not define a clean interface and also have practical implementation problems.

The general implementations of X.25 applications on hosts invoke appropriate processes based on CUD fields received in CALL packet, e.g., login process on receipt of (1,0,0,0), SNDCF on 0xCC, etc. The available X.25 packages generally give users the options to add more such applications over the X.25 drivers, by simply defining new CUD fields and mapping these to the new application. Only configuration additions are required, which can be performed even by end users. A data pattern based invocation of an application is impractical in this scenario, without a rewrite of the software.

Further, the usual method of running the login process on any OS is by invoking appropriate pseudo-terminal login driver to run over the SVC created after call establishment. The receipt of PAD PID causes the "login" driver to be invoked. A data pattern based approach will require this driver to interpret the data and invoke PPP process from this driver, instead of continuing with the login process. This is generally impractical, since one may need to modify standard OS login drivers and also invocation of another process from within a driver is generally not possible in different OSes.
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 two 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 mechanisms layer above it.

The proposed mechanisms software 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 frame, 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

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 frame has been received as multiple X.25 packets.

   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 an 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. Acknowledgements

The author is grateful to Jonathan Goodchild for his suggestions on the choice of CUD.

5. References


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