WANPIPE™

Working with WANPIPE in Linux®

CONFIGURATION M A N U A L

Author: Nenad Corbic

Introduction

WANPIPE from Sangoma Technologies Inc. (http://www.sangoma.com) is a family of intelligent multi-protocol WAN adapters with data transfer rates up to 4Mbps. They are also known as Synchronous Data Link Adapters (SDLA) and are designated as S514-PCI or S508-ISA. These cards support

- X.25, Frame Relay, PPP, Cisco HDLC protocols.

- API support for protocols like HDLC (LAPB),
  HDLC Streaming, X25, Frame Relay and BiSync.

- X25 PAD Support.

- Ethernet Bridging over Frame Relay protocol.

- MULITLINK PPP, used for bundling multiple T1/56K dialup
  lines into to a single logical link.

- Async TTY serial port on S514-PCI/S508-ISE Secondary Port.
  The async port can replace a standard UART serial port, and interface to
  a modem to support a backup dial up connection.

Sangoma S514 PCI and S508 ISA are intelligent adaptors that support most WAN protocols in firmware. The fact that protocols are supported in firmware makes device driver design much simpler.

Driver's major responsibility is to pass data between the adaptor and Linux operating system. Furthermore, by using the intelligent adapters, system CPU load is kept at the minimum, an important factor in Server-Router performance and stability: a relatively slow machine, like a 486, can be used with the Sangoma adapter and Linux OS to create a powerful T1 router/firewall.

Another benefit of isolating protocols, on the board, is the possibility to test and certify protocol implementations on one operating system and be sure that they will work identically on any other operating system. If necessary, a protocol update can be installed on the fly, without recompiling the driver or the kernel.

Sangoma Adapters can have two different physical interfaces, T1 (CSU/DSU on board) or serial V.35/X.21/EIA530/RS232. The card with the T1 interface allows the server to connect directly to the T1 line without an external CSU/DSU.

WANPIPE Configuration

The WANPIPE configuration process starts with creating a detailed configuration file that outlines the hardware, protocol and IP options as well as location of the adapter firmware.   It is created for each WANPIPE device.

> A WANPIPE device does not describe a physical card, but a logical implementation of a physical card. For example, a S5141 card contains a single CPU with two physical ports: a High-Speed (up to 4Mbps) port and a Low-Speed (up to 512Kbps) port. Furthermore, each port can support an independent physical connection. Therefore, to reiterate, *number of Wanpipe devices refers to the number of physical lines connected to a Sangoma adapter.*

To simplify the WANPIPE configuration process, a GUI configuration utility called **wancfg** was developed. It is located in /usr/local/wanrouter directory.

```
Ex: cd /usr/local/wanrouter
    ./wancfg
```

NOTE: wancfg has extensive help files for each WANPIPE option.

If wancfg cannot be used due to lack of bash2 support, please use the sample Wanpipe configuration files to aid you in configuring WANPIPE. The sample configuration files are located in /usr/local/wanrouter/samples directory.

Frame Relay

Frame Relay is a simplified form of Packet Switching similar in principle to X.25 in which synchronous frames of data are routed to different destinations depending on header information.

The biggest difference between Frame Relay and X.25 is that X.25 guarantees data integrity and network managed flow control at the cost of some network delays. Frame Relay switches packets end to end much faster, but **there is no guarantee of data integrity at all.**

Frame Relay is cost effective, partly due to the fact that the network buffering requirements are carefully optimized. Compared to X.25, with its store and forward mechanism and full error correction, **network buffering is minimal**. Frame Relay is also **much faster than X.25**: the frames are switched to their destination with only a few byte times delay, as opposed to several hundred milliseconds delay on X.25.

Wanpipe Frame Relay Options

Wanpipe Frame Relay has the following modes of operations: WANPIPE, API and BRIDGING.

WANPIPE MODE: The Linux Kernel uses Frame Relay logical channels to route packets to remote networks, using the TCP/IP protocol.

Each logical channel is represented by a network interface, where each interface contains unique IP information. Kernel uses the IP information to route incoming packets to remote networks.

API MODE: Frame Relay API mode is used to send non-IP traffic over a frame relay link. The API interface allows the user to build a custom application on top of the frame relay link in order to transmit custom data packets (i.e Non IP). An example of a custom application would be Voice-over IP, Data capture and packet analysis.

BRIDGING MODE: The 'kernel bridge' is used to bridge multiple frame relay logical channels together into a single LAN. This option is desirable if IP addresses are scarce, or in building a single LAN architecture.   Thus, multiple remote LAN's can be

bridged together into a single LAN using the frame relay (WAN) links.

Please refer to WanpipeEthernetBridge.(pdf/txt) for further information.

Information needed from your ISP

- List of DLCI (channels) used:
    DLCI is a logical Frame Relay link/channel (16 – 4096).

- IP address for each DLCI channel (WANPIPE MODE)
    Ex: Local: 201.1.1.1
         Remote: 201.1.1.2

- Clocking Mode:
    In most cases clocking will be External.
    i.e. the ISP will supply the clock.

- Frame Relay Signalling:
    Frame relay has number of singalling options:
        LMI, ANSI, Q933

- Frame Relay Station:
    Frame relay has two modes of operation:
        CPE or NODE.

    - CPE: customer premises equipment. As an end user, a Frame Relay connection should always be set to this mode.

    - NODE: switch emulation:
        This option should only be used in back-to-back test situation,   with two sangoma card. Sangoma can act as a switch, however, in most cases that is an ISP's job.

- CSU/DSU Configuration:
    Sangoma S514/S508FT1 cards are supplied with an onboard CSU/DSU that needs to be configured, based on the type line it's being connected to.

Cisco HDLC

Cisco HDLC is a point-to-point protocol implemented on top of HDLC layer 2. As the name implies CHDLC is a protocol mostly used to connect to the Cisco^{tm} external routers.

Wanpipe CHDLC Options

Wanpipe CHDLC has the following modes of operations: WANPIPE and API.

WANPIPE MODE: The Linux Kernel uses the CHDLC point-to-point link to route packets to a remote network, using the TCP/IP protocol.

A CHDLC point-to-point connection is represented by a single network interface that contains IP information obtained from the ISP. The Kernel uses the IP information to route incoming packets to remote a network.

API MODE: CHDLC mode used to send non-IP traffic over a CHDLC point-to-point link. The API interface allows the user to build a custom application on top of the CHDLC link in order to transmit custom data packets (i.e Non IP). An example of a custom application would be a Satellite Receive Only data collector or Data capture and packet analysis tool.

Information needed from your ISP

- IP address for the CHDLC point-to-point connection (WANPIPE MODE)
      Ex: Local: 201.1.1.1
            Remote: 201.1.1.2

- Clocking Mode:
      In most cases clocking will be External.
      i.e. the ISP will supply the clock.

- CSU/DSU Configuration:
      Sangoma S514/S508FT1 cards are supplied with an onboard CSU/DSU that needs to be configured, based on the type line it's being connected to.

X25

X.25 Packet Switched networks allow remote devices to communicate with each other across high-speed digital links without the expense of individual leased lines. Packet Switching is a technique whereby the network routes individual packets of HDLC data between different destinations based on addressing within each packet

The protocol known as X.25 encompasses the first three layers of the **OSI 7-layered architecture** as defined by the International Organization for Standardization (ISO) as follows:

- **Layer 1: The Physical Layer** is concerned with electrical or signaling. It includes several standards such as V.35, RS232 and X.21.
- **Layer 2: The Data Link Layer**, which is an implementation of the ISO HDLC standard called Link Access Procedure Balanced (LAPB) and provides an error free link between two connected devices.
- **Layer 3: The Network Layer,** which provides communications between devices connected to a common network. In the case of X.25, this layer is referred to as the **X.25 Packet Layer Protocol (PLP)** and is primarily concerned with network routing functions and the multiplexing of simultaneous logical connections over a single physical connection.

Wanpipe X25 Options

Wanpipe X25 has the following modes of operations: WANPIPE, API.

WANPIPE MODE: The Linux Kernel uses X.25 logical channels to route packets to remote networks, using the TCP/IP protocol.

Each logical channel is represented by a network interface, where each interface contains unique IP information. Kernel uses the IP information to route incoming packets to remote networks.

API MODE: X.25 API mode is used to send non-IP traffic over an X.25 link. The API interface allows the user to build a custom application on top of the X.25 link in order to transmit custom data packets (i.e Non IP). An example of a custom application would be Credit Card verification, Data capture and packet analysis.

X25 PAD: X25 PAD

Information needed from your ISP

- List of LCN (logical channel number) used:
    X25 LCN's can be configured as:

    - Switched Virtual Circuits (SVC)
        SVC is analogous to a telephone line. A call must be established before communication takes place.

    - Permanent Virtual Circuits (PVC)
        PVC line is always connected, thus not calls setup is required.

    The ISP must provide LOWEST (SVC/PVC) and HIGHEST (SVC/PVC) numbers.

- IP address for each LCN channel (WANPIPE MODE only)
    Ex: Local: 201.1.1.1
        Remote: 201.1.1.2

- Clocking Mode:
    In most cases clocking will be External.
    i.e. the ISP will supply the clock.

- X25 Station:
    X25 has two modes of operation:
        DCE or DTE.

PPP

Point-To-Point Protocol (PPP) is a protocol implemented on top of HDLC layer 2.PPP is a standard protocol used when connecting over a point-to-point link.

Wanpipe PPP Options

Wanpipe PPP has the following modes of operations: WANPIPE.

WANPIPE MODE: The Linux Kernel uses the PPP point-to-point link to route packets to a remote network, using the TCP/IP protocol.

A PPP point-to-point connection is represented by a single network interface that contains IP information obtained from the ISP. The Kernel uses the IP information to route incoming packets to remote a network.

API MODE: Not-Supported.

Information needed from your ISP

- IP address for the PPP point-to-point connection (WANPIPE MODE)
       Ex: Local: 201.1.1.1
            Remote: 201.1.1.2

- Clocking Mode:
       In most cases clocking will be External.
       i.e. the ISP will supply the clock.

- CSU/DSU Configuration:
       Sangoma S514/S508FT1 cards are supplied with an onboard CSU/DSU that needs to be configured, based on the type line it's being connected to.

Important

This version of PPP protocol is supported by WANPIPE in firmware, as with all other protocols.   The limitation of the current WANPIPE PPP protocol is that it cannot run on a Secondary port of the Sangoma adapter (ex. S514-1).

Multi-Port Sync PPP

The Multi-Port Sync PPP is a standard implementation of PPP protocol implemented in the Linux Kernel, NOT in WANPIPE firmware. Using the Sangoma adapter as a dumb card and the Linux PPP Sync Layer a standard PPP connection can be established over a T1 Link.

The Multi-Port PPP has been developed to address the standard WANPIPE PPP limitation. The standard WANPIPE PPP (supported in firmware) cannot run on a Secondary port. Since the Multi-Port PPP is implemented in kernel, second port is freed up; thus, multiple independent PPP connections can be established on both Sangoma adapter ports simultaneously.

Wanpipe Multi-Port Sync PPP Options

Wanpipe Multi-Port PPP has the following modes of operations: WANPIPE.

WANPIPE MODE: The Linux Kernel uses the PPP point-to-point link to route packets to a remote network, using the TCP/IP protocol.

A PPP point-to-point connection is represented by a single network interface that contains IP information obtained from the ISP. The Kernel uses the IP information to route incoming packets to remote a network.

API MODE: Not-Supported.

Information needed from your ISP

- IP address for the PPP point-to-point connection
  (WANPIPE MODE)
      Ex: Local: 201.1.1.1
          Remote: 201.1.1.2

- Clocking Mode:
      In most cases clocking will be External.
      i.e. the ISP will supply the clock.

- CSU/DSU Configuration:
      Sangoma S514/S508FT1 cards are supplied with an onboard CSU/DSU that needs to be configured, based on the type line it's being connected to.

Sync/Async/Multilink (TTY) PPP

The Sync/Async/Multilink PPP is a standard implementation of the PPP protocol implemented in the Linux Kernel, NOT in WANPIPE firmware.   Using the Sangoma adapter as a TTY serial card and the Linux PPP Layer a standard PPP connection can be established over a T1 Link or a Modem line.

The Sync/Async/Multilink PPP has been developed to address the standard WANPIPE PPP limitations:
- The standard WANPIPE PPP (supported in firmware) cannot run on a Secondary port and
- Does not support Multilink operation.

The Sync/Async/Multilink PPP has two modes of operation:
- Sync with Multilink option.
  - Used to establish a PPP connection over sync T1 lines.
- Async.
  - Use to establish a PPP connection via MODEM over a telephone line.

NOTE: Both modes are using for ROUTING purposes, i.e. no API support.

Furthermore, since the Sync/Async PPP is implemented in the kernel, the second port is freed up; thus, multiple independent PPP connections can be established on both Sangoma adapter ports simultaneously.

SYNC Mode

Using the PPPD daemon, kernel Sync-PPP layer and the Wanpipe sync TTY driver: a PPP protocol connection can be established via Sangoma adapter, over a T1 leased line.

The 2.4.0 kernel PPP layer supports MULTILINK protocol. It can be used to bundle any number of Sangoma adapters (T1 lines) into one, under a single IP address. Thus, efficiently obtaining multiple T1 throughputs.

NOTE: The remote side must also implement MULTILINK PPP protocol.

ASYNC Mode

Using the PPPD daemon, kernel Async PPP layer and the WANPIPE async TTY driver: a PPP protocol connection can be established via Sangoma adapter and a modem, over a telephone line.

The WANPIPE Async TTY driver simulates a serial TTY driver that is normally used to interface the MODEM to the Linux kernel.

NOTE: This option only works on a SECONDARY Port of the S514-PCI/S508-ISA card.

Device /dev/ttyW(0,1,2..)

To interface a PPPD daemon to the WANPIPE TTY driver a /dev/ttyWX X={0,1,3...) device must be created.

ex: mknod -m 666 /dev/ttyW0 c 226 0

Note: 226 is the Major Number
      0   is the Minor Number

IMPORTANT:
This option should only be used if the MULTILINK option desired, to bundle T1 connections together or to simulate a serial async device driver.   Otherwise, it is recommended that standard WANPIPE PPP is used.

Information needed from your ISP

- IP address for the PPP point-to-point connection
  If STATIC IP addressing is used.
    Ex: Local: 201.1.1.1
        Remote: 201.1.1.2

- Clocking Mode:
    In most cases clocking will be External.
    i.e. the ISP will supply the clock.

- CSU/DSU Configuration:
    Sangoma S514/S508FT1 cards are supplied with an onboard CSU/DSU that needs to be configured, based on the type line it's being connected to.

Sync/Async TTY PPP Configuration and Operation

1. The WANPIPE TTY driver has very few options since main configuration options will be defined during the PPPD daemon configuration.

2. WANPIPE TTY OPTIONS:
   (Use the **wancfg** to create the wanpipe1.conf configuration files.)

   TTY_MINOR: TTY MINOR represents a TTY port.
                        Options: 0,1,2,3 … Default: 0

                        It binds a WANPIPE device driver to the /dev/ttyWX device , where X=(0,1,3,..).   For example MINOR number 0 binds a Wanpipe TTY driver to /dev/ttyW0. Thus, when pppd daemon opens the /dev/ttyW0 it will reach the device driver whose Minor number is 0.

                        Note TTY_MINOR must differ for each Wanpipe device.

   TTY_MODE: WANPIPE TTY driver operation mode
                        Options: Sync or Async   Default: Sync

                        The driver operation mode must be specified here, since driver cannot obtain the operation mode from the pppd configuration calls. Therefore, if the driver operation mode is Sync than the pppd must be invoked with the sync option.

   Furthermore, all subsequent drivers must be configured with the same TTY MODE and different MINOR numbers.

3. The **wancfg** configurator will configure the pppd daemon according to the TTY_MODE selected.   It will also prompt the user for MULTILINK support. Three files will be created for each Wanpipe device:
   - /etc/ppp/options.ttyWX   X is TTY_MINOR number.
   - /etc/ppp/peers/isp_wanpipeX X is a device number (1-16)
   - /dev/ttyWX   X is TTY_MINOR number.

4. WANPIPE TTY drivers must be started before the pppd attempts to open a /dev/ttyWX device.
   ```
   ex: wanrouter start wanpipe1
   ```

5. Once the Wanpipe device is started, PPP connection can be established by calling the pppd call script (created by wancfg):
   ```
   ex: pppd call isp_wanpipe1
   ```

Manual PPPD Daemon Configuration

1. Depending on the TTY MODE used, the pppd configuration file must be configured for Sync or Async operation.

   The pppd daemon uses an options configuration file found in /etc/ppp directory. An options configuration file exists for each /dev/ttyX device. For /dev/ttyW0 device, the following options file must be created in /etc/ppp directory:

   Async PPPD daemon configuration:

   /etc/ppp/options.ttyW0

   ```
   #Example options.ttyW0 file
   asyncmap 0
   modem        #Use the modem signals

   silent       #Wait until ppp request is received before
                #starting ppp protocol (optional).

   Debug        #Eanble debugging
                (optional)

   Crtscts      #Eanble crtscts hardware flow control

   noipdefault  #Use dynamic IP addressing. Obtain IP
                #addresses from the remote side.
                #NOTE: disable this option if using STATIC IP
                       addressing.  (async only)

   --detach     #Do not detach from the terminal window

   defaultroute #IP address of this interface should be set as
                #default in the routing table.

   #End of options.ttyW0 file
   ```

Sync PPPD Daemon configurtion:

/etc/ppp/options.ttyW0

```
#Example options.ttyW0 file
asyncmap 0
silent          #Wait until ppp request is received before
                #starting ppp protocol (optional).

Debug           #Eanble debugging
                (optional)

--detach        #Do not detach from the terminal window

defaultroute    #IP address of this interface should be set as
                #default in the routing table.

#End of options.ttyW0 file
```

1. Furthermore a call pppd script can also be defined to simplify the pppd argument line. The call script must be defined in /etc/ppp/peers directory. The example call script will be called **isp_async** and **isp_sync**:

Async PPP call script

```
/etc/ppp/peers/isp_wanpipe1:
        ttyW0                   #Wanpipe TTY driver
        38400                   #Baud Rate
        connect '/etc/ppp/redialer'  #A modme dial up script
```

NOTE: In async mode, WANPIPE TTY drivers are always set to internal clocking, thus the baud rate needs to be set here. The driver obtains the baud rate through pppd configuration calls, not wanpipe1.conf configuration file as in sync.

Sync PPP call script

```
/etc/ppp/peers/isp_wanpipe1:
        ttyW0           #Wanpipe TTY driver
        sync            #Connect via sync line
        <IP local>:<IP remote>     #Set to IP addresses obtained
                                   # from ISP.
```

NOTE: The baud rate is not needed since the Sync Wanpipe TTY drivers obtain the baud rate from the Wanpipe configuration files (ex. wanpipe1.conf).

Starting the pppd daemon using the above script:
```
Ex: pppd call isp_wanpipe1
```


Multilink PPPD Configuration

One of the major reasons for WANPIPE TTY driver development was
MULTILINK PPP.   The 2.4.X kernels support this protocol, which can bundle
multiple WANPIPE T1 cards into a singe logical connection to achieve greater
throughput.   MULTILINK PPP protocol can be used in Sync or Async mode.
The following configuration changes need to be applied to the above pppd call
scripts in order to support mulitlink.

The /etc/options.ttyWX files do not change only the call scripts do:

Async PPP call script, MULTILINK support:

```
/etc/ppp/peers/isp_wanpipe1
        #Configuration for the first 56K line
        ttyW0                   #First Wanpipe device
        38400                   #Baud Rate
        multilink               #Enable Multilink support
        connect '/etc/ppp/redialer'  #A modme dial up script

/etc/ppp/peers/isp_wanpipe2
        #Configuration for the second 56K line
        ttyW1                   #Second Wanpipe device
        38400                   #Baud Rate
        multilink               #Enable Multilink support
        noip                    #Slaves have no IP info
        connect '/etc/ppp/redialer'  #A modem dial up script
```

Sync PPP call script, MULTILINK support:

```
/etc/ppp/peers/isp_wanpipe1
        #Configuration for the first T1 line
        ttyW0           #First Wanpipe device
        sync            #Connect via sync line
        multilink       #Enable Multilink Protocol
        <IP local>:<IP remote>     #Set to IP addresses obtained
                                   # from ISP.
```

```
/etc/ppp/peers/isp_wanpipe2
        #Configuration for the second T1 line
        ttyW1           #Second Wanpipe device
        sync            #Connect via sync line
        multilink       #Enable Multilink Protocol
        noip            #Slaves have no IP info.
```

To start the pppd daemon and bundle the two links together do the following:

```
Ex: pppd call isp_wanpipe1
```

Once the first ppp connection comes up then start the second.

```
Ex: pppd call isp_wanpipe2
```