IP link Quantar V.24 systems using Cisco routers

This thread was published on the P25.CA / Communications.support web site on July 2012

© John Yaldwyn ZL4JY

In the thread:

https://communications.support/threads/1856-Exploring-the-Quantar-V-24-modem-interface

MattSR and I explored the Quantar protocol commonly called V.24 in the context of the RT-RT repeater to repeater configuration. At that time I was looking at building a V.24 to IP adapter based on a low cost 32 bit ARM microcontroller development board, a project dubbed QuantiPHY. Easy to get hardware, hard to get going, and heaps of software work.

In the thread:

https://communications.support/threads/4214-Link-your-Quantar-over-IP-with-some-crusty-old-modems-for-CHEAP

mizzotch introduced the concept of using V.32 phone line modems linked over low cost VOIP hardware from Obi using the ObiTalk network. This linking approach works amazing well. Mars, mizzotch, and I have all linked Quantar machines together this way and the results have been great, hats off to mizzotch for his lateral thinking. Easy to get hardware, easy to get going, and no software work.

Unfortunately, the full DIY approach was taking too much time while the ObiTalk approach has some issues with latency and continuous operation. While experimenting with these two schemes it occurred to me that there was a third option meeting the key ‘easy to get, low cost hardware’ requirement and needing only some moderate configuration work while offering significant advantages over the other two methods. The work done on the first two options was time well spent as it made clear what was required for an effective V.24 linking solution. Using off-the-shelf routers is not a new idea, there is a couple of threads here and on the Batboard that reference Cisco but with few details specific to the Motorola V.24 synchronous interface needed for P25 mode.

Cisco Systems have been making routers since the mid-eighties. Wikipedia says of Cisco “it was one of the first to sell commercially successful routers supporting multiple network protocols”. And that’s the key to this story, the ability of Cisco’s router products to work with a variety of protocols.

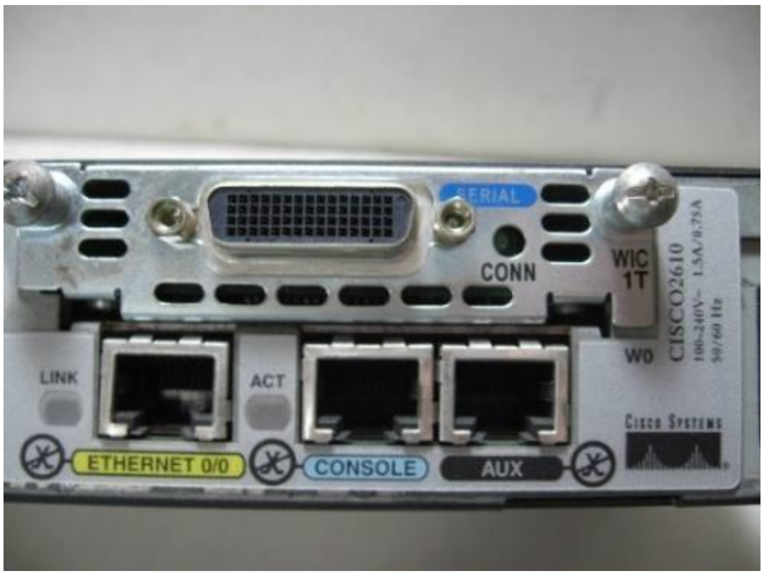
This thread explains how to link Quantar stations together over IP for P25 digital only operation using off-the-shelf Cisco hardware. Surplus Cisco routers can be easily found, typically listing on eBay in the $50 to $100 range. As with the 'Exploring the Quantar V.24 Interface' thread, this will be a series of postings that will detail various aspects of how to do this as I’ve not found anywhere else implementation details for the Motorola V.24 synchronous interface over Cisco products. It might be an idea to hold off on questions until I’ve had the chance post the series. So let’s get busy. This is a typical single rack unit high Cisco router, it's boring as all the good stuff is around the back:

I’m going to focus on a few end-of-life (EOL) Cisco routers, this doesn’t mean that other earlier or later Cisco hardware won’t do the job but as this is for recreational interest, budget and ease of use will be the biggest drivers. In particular the Cisco 2600XM and 3700 series rack-mountable modular multiservice platforms offer a huge bang for buck on the used market. The advantages of these models are that they’re plentiful while still being new enough to be reliable and they avoid some of the snags of the older designs like the 2500 series. While the 2600 came out in the late nineties, I’m using the newer 2600XM series in particular. Surplus units were typically made in 2003/2004 and the last date of support from Cisco for this series was last year. Today people are using the much newer 2811 and 1841 router.

Corporates IT departments don’t usually hold on to equipment later than EOL, dumping otherwise good kit as part of upgrade programs. Here's a stack of routers purchased from an on-line auction, top to bottom are three 2600XM routers each with one WIC 1T interface for sync V.24 (one unit has a T1 interface as well), a 3725 router with four serial interfaces, finally another 3725 with eight serial interfaces and a 16 port Ethernet switch option:

I would avoid the 2600 series without the XM suffix unless you are certain of the memory configuration. Lots of sellers simply describe these routers as 'Cisco 2600' but there are single and dual Ethernet options with 10 and 100 Mbps versions, see the Cisco reference link at the bottom of this post. You want a 2610XM as a minimum; it's more capable brothers like the 2611XM, 2620XM, and 2612XM, 2651XM are all good.

Here’s a close up of what not to buy, a plain 2610 with a WIC 1T serial card, note the X in a circle next to the Ethernet, Console, and Aux ports:

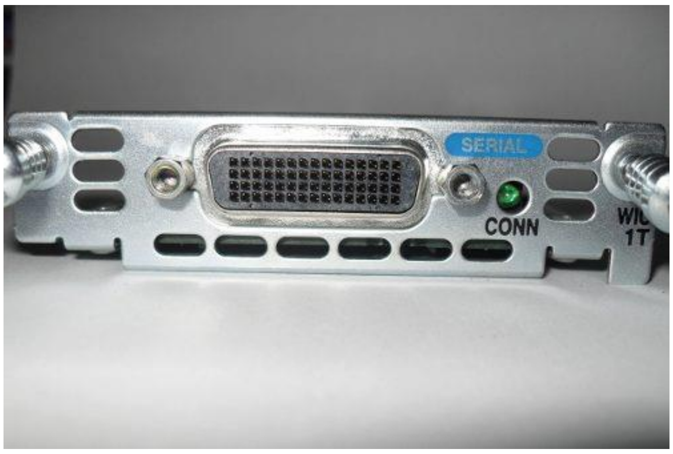


Compare that to the XM versions in my second image. X inside the circle = avoid. IMHO, YMMV, memory dependent, offer not available in all stores.

Don’t touch the older 2500 and 4000 series. Also avoid the 1750 access routers, except perhaps the 1751. The 1760 and 3800 series might be worth a look, but I have no experience with these. Newer equipment like the 1800 and 3800 is nice but still pricey even used as they are favoured by IT folk studying for Cisco certification.

For this project the minimum memory needed will be 16MB of Flash and 64MB of RAM. You will need a router that has a WIC-1T serial card to connect to the Quantar V.24 / RS-232 synch interface.

Here’s one naked:

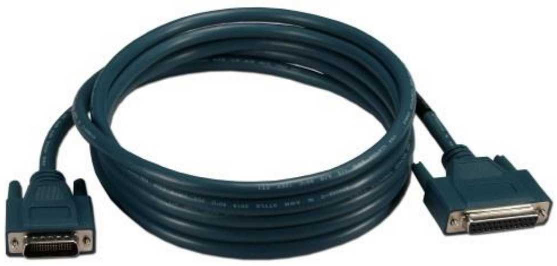


These are easily recognized by the high density 60 pin Molex LFH series connector (low force helix, if you really want to know). Best to get a router with the right interface and memory already fitted as they can sell for silly money parted out on their own. Don't pay more than $50 to $75 for a router and $10 to $20 (if you have to) for a serial card. Also look for a console cable, they can often be had for the asking as one comes with every router shipped by Cisco:

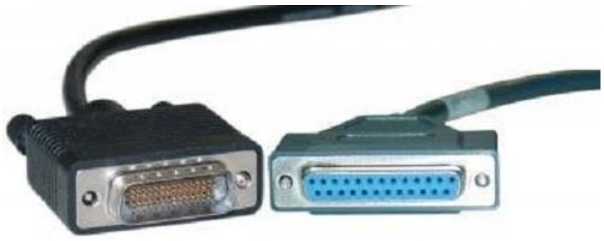
Next we'll talk about the Cisco IOS (internetwork operating system) and powering on your first router!

Reference: http://www.cisco.com/c/en/us/products/collateral/routers/2600-series-multiservice-platforms/product_data_sheet0900aecd800fa5be.html

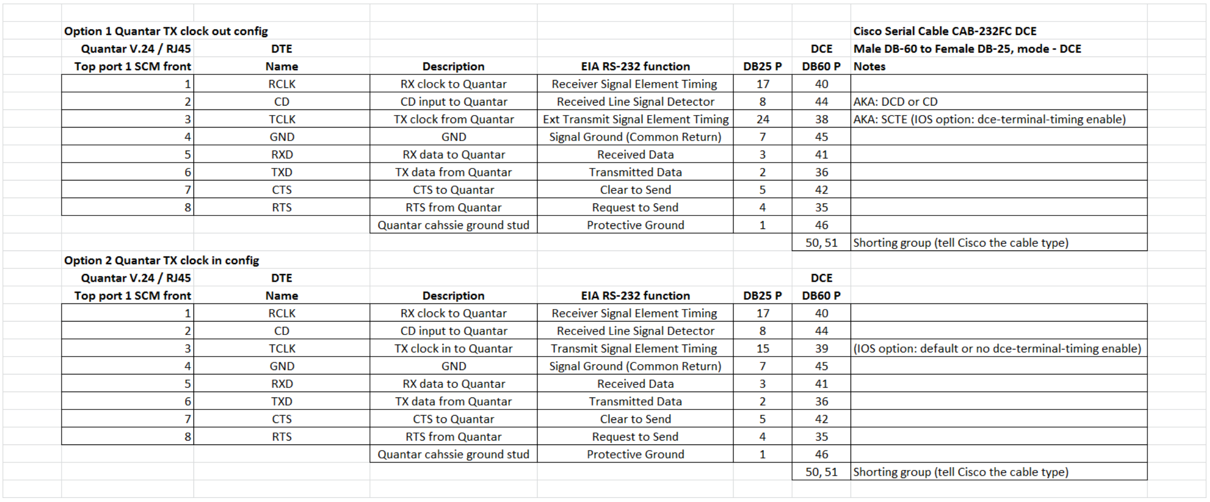
As a postscript to the above post, when Cisco shopping keep an eye out for the necessary RS-232 cables. The Cisco CAB-232FC serial cable, part number 72-0794-01, is a male DB-60 to female DB-25 assembly. The cable has 'Cisco' and 'DCE' moulded into the connector bodies. Here's what it looks like:



The detail of the connector ends are as shown, don't get the DTE cable with the male DB25:



And here are the details for RJ-45 to DB25 adapter cable needed between the Quantar and the end of the Cisco cable:



Option 1 Quantar TX clock out config						Cisco Serial Cable CAB-232FC DCE
Quantar V.24 / RJ45	DTE				DCE	Male DB-60 to Female DB-25, mode - DCE
Top port 1 SCM front	Name	Description	EIA RS-232 function	DB25 P	DB60 P	Notes
1	RCLK	RX clock to Quantar	Receiver Signal Element Timing	17	40	
2	CD	CD input to Quantar	Received Line Signal Detector	8	44	AKA: DCD or CD
3	TCLK	TX clock from Quantar	Ext Transmit Signal Element Timing	24	38	AKA: SCTE (IOS option: dce-terminal-timing enable)
4	GND	GND	Signal Ground (Common Return)	7	45	
5	RXD	RX data to Quantar	Received Data	3	41	
6	TXD	TX data from Quantar	Transmitted Data	2	36	
7	CTS	CTS to Quantar	Clear to Send	5	42	
8	RTS	RTS from Quantar	Request to Send	4	35	
		Quantar cahssie ground stud	Protective Ground	1	46	
					50, 51	Shorting group (tell Cisco the cable type)

Option 2 Quantar TX clock in config						
Quantar V.24 / RJ45	DTE				DCE	
Top port 1 SCM front	Name	Description	EIA RS-232 function	DB25 P	DB60 P	
1	RCLK	RX clock to Quantar	Receiver Signal Element Timing	17	40	
2	CD	CD input to Quantar	Received Line Signal Detector	8	44	
3	TCLK	TX clock in to Quantar	Transmit Signal Element Timing	15	39	(IOS option: default or no dce-terminal-timing enable)
4	GND	GND	Signal Ground (Common Return)	7	45	
5	RXD	RX data to Quantar	Received Data	3	41	
6	TXD	TX data from Quantar	Transmitted Data	2	36	
7	CTS	CTS to Quantar	Clear to Send	5	42	
8	RTS	RTS from Quantar	Request to Send	4	35	
		Quantar cahssie ground stud	Protective Ground	1	46	
					50, 51	Shorting group (tell Cisco the cable type)

See the Cisco reference for more detail. I'll explain the Option 1 vs Option 2 difference later.

NOTE: the table refers to the top V.24 port on the Quantar, this should be the bottom V.24 port if you are using a real Motorola TTN4010 W/L daughter board.

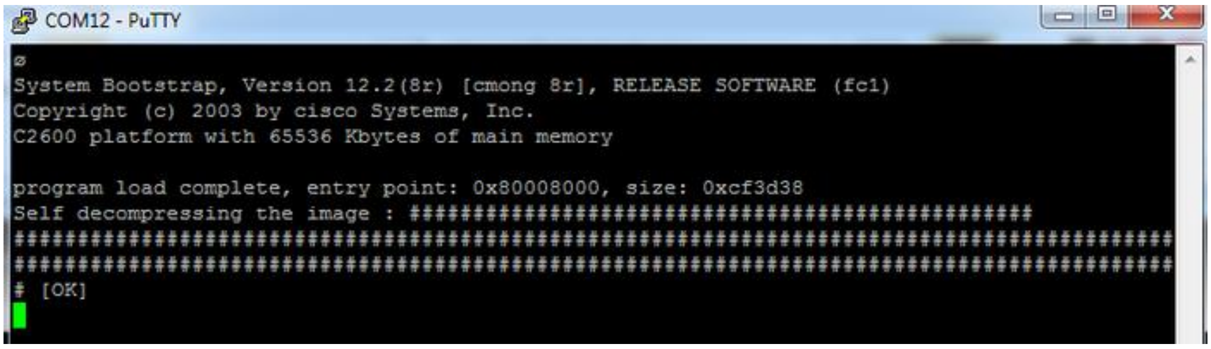
Reference:

http://www.cisco.com/c/en/us/support/docs/routers/10000-series-routers/46800-cab232mtfc.html#cab-232fc

Ok so you have your router(s) and its time power on.

You'll need a serial terminal program like HyperTerminal, but that's now history if you're using Vista or Windows 7. The best option today is Simon Tatham's PuTTY, which can download from Reference 1 at the end of this post. PuTTY works with both a serial connection from your PC (a USB serial com port is fine) and supports Telnet/SSH over Ethernet, which you'll probably want later.

Hook up your blue console cable to the router's RJ45 Console port with the DB9 to the PC. Don't turn on the router just yet. Launch PuTTY , click the serial button, enter the relevant COM port and 9600 bps, and then hit open. The result should be a black window, so go ahead now and turn on the router. You should see something very much like this:



```
ø
System Bootstrap, Version 12.2(8r) [cmong 8r], RELEASE SOFTWARE (fc1)
Copyright (c) 2003 by cisco Systems, Inc.
C2600 platform with 65536 Kbytes of main memory

program load complete, entry point: 0x80008000, size: 0xcf3d38
Self decompressing the image : ################################################
##########################################################################
##########################################################################
# [OK]
```

So from the output we find out how much RAM memory the router has (65536 Kbytes = 64MB), then after a short wait the router takes the packed system software from flash and unzips it to memory.

After this excitement it gets a bit tedious but there is still a lot of useful information in the boot output, let's review:

```
System Bootstrap, Version 12.2(8r) [cmong 8r], RELEASE SOFTWARE (fc1)
Copyright (c) 2003 by Cisco Systems, Inc.
C2600 platform with 65536 Kbytes of main memory

program load complete, entry point: 0x80008000, size: 0xcf3d38
Self decompressing the image :

##################################################
################################################## ################
##################################################
################################################## ################
################ [OK]

Smart Init is enabled
Smart init is sizing iomem
ID MEMORY_REQ TYPE
00036A 0X000BA600 C2610XM Single Fast Ethernet
0X000F3BB0 public buffer pools
0X00211000 public particle pools
TOTAL: 0X003BF1B0

If any of the above Memory Requirements are
"UNKNOWN", you may be using an unsupported
configuration or there is a software problem and
system operation may be compromised.
Rounded IOMEM up to: 4Mb.
Using 6 percent iomem. [4Mb/64Mb]

Restricted Rights Legend

Use, duplication, or disclosure by the Government is
subject to restrictions as set forth in subparagraph
(c) of the Commercial Computer Software - Restricted
Rights clause at FAR sec. 52.227-19 and subparagraph
(c) (1) (ii) of the Rights in Technical Data and Computer
Software clause at DFARS sec. 252.227-7013.

cisco Systems, Inc.
170 West Tasman Drive
San Jose, California 95134-1706



Cisco Internetwork Operating System Software
IOS (tm) C2600 Software (C2600-A3JK9S-M), Version 12.2(27), RELEASE SOFTWARE
(fc3)
Copyright (c) 1986-2004 by cisco Systems, Inc.
Compiled Tue 02-Nov-04 23:43 by kellmill
Image text-base: 0x8000808C, data-base: 0x81732764

cisco 2610XM (MPC860P) processor (revision 0x200) with 61440K/4096K bytes of
memory.
Processor board ID JAE08020YTD (3231645583)
M860 processor: part number 5, mask 2
Bridging software.
X.25 software, Version 3.0.0.
SuperLAT software (copyright 1990 by Meridian Technology Corp).
TN3270 Emulation software.
1 FastEthernet/IEEE 802.3 interface(s)
```

```
1 Serial network interface(s)
32K bytes of non-volatile configuration memory.
16384K bytes of processor board System flash (Read/Write)



Press RETURN to get started!


00:00:13: %LINK-3-UPDOWN: Interface FastEthernet0/0, changed state to up
00:00:13: %LINK-3-UPDOWN: Interface Serial0/0, changed state to down
*Mar 1 00:00:14.559 UTC: %LINEPROTO-5-UPDOWN: Line protocol on Interface
FastEthernet0/0, changed state to down
*Mar 1 00:00:14.559 UTC: %LINEPROTO-5-UPDOWN: Line protocol on Interface
Serial0/0, changed state to down
*Mar 1 00:00:15.276 UTC: %SYS-5-CONFIG_I: Configured from memory by console
*Mar 1 00:00:19.112 UTC: %SYS-5-RESTART: System restarted --
Cisco Internetwork Operating System Software
IOS (tm) C2600 Software (C2600-A3JK9S-M), Version 12.2(27), RELEASE SOFTWARE
(fc3)
Copyright (c) 1986-2004 by cisco Systems, Inc.
Compiled Tue 02-Nov-04 23:43 by kellmill
*Mar 1 00:00:19.112 UTC: %SNMP-5-COLDSTART: SNMP agent on host P25R1 is
undergoing a cold start
*Mar 1 00:00:20.132 UTC: %LINEPROTO-5-UPDOWN: Line protocol on Interface
FastEthernet0/0, changed state to up
Interface FastEthernet0/0 assigned DHCP address 192.168.1.202, mask 255.255.255.0

P25R1 con0 is now available

Press RETURN to get started.
```

OK so what we have here is information relating to the memory allocation, which we can ignore as in most cases the router will be set to manage this itself, followed by the Restricted Rights Legend, and the vital bit for us namely the router software feature set 'C2600-A3JK9S-M', and the version '12.2(27)', (show version C2600-A3JK9S-MZ.122-27.bin).

Cisco organize their software releases to include various feature sets, from the basic IPbase up to the most feature rich Advanced Enterprise Services. The more features you want the bigger the IOS image becomes. Low cost routers often have limited memory, less than 64MB is needed for IPbase, and more than 192MB is needed for reasonably modern Advanced Enterprise Services. For more detail see Reference 2.

Earlier I said that a minimum of 16MB of Flash and 64MB of RAM was needed for this project. That's because we need a minimum feature set of Enterprise Base. While IOS 12.2 is not very modern, it is adequate for the task and more importantly small enough to fit in cheap memory constrained routers. Next in the output we have details of the hardware platform, in this case a 2610XM, and finally a series of status messages regarding the various interface states as the configuration of the router is executed. Today something like 15.1 would be a good choice for newer hardware.

Try some simple commands like <`show version`> (mentioned earlier) or <`show diag`>.

Now ideally you want to buy a router that has already has Enterprise Base, Enterprise Services, or perhaps even Advanced Enterprise Services loaded. And that's where things get slightly tricky. Cisco provide a non-transferable licence to the firmware in the router to the original purchaser, you are supposed to re-licence the software when you buy second-hand. However, I've never seen a second hand Cisco router sold without firmware and a wide range of Cisco firmware images are all over the 'net. Technicians and engineers seem to rely on these to study for Cisco certification. Routers are easy to update as no 'special tools' are required. Large commercial users are unlikely to base their network on eBay routers. In any case for commercial users Cisco offer a remarkably good deal called SMARTnet, it a maintenance contract that covers 24/7 support, hardware replacement options (even for you beat up eBay router), IOS software updates, etc. A good deal from a great company.

There are three candidate protocols that could be used to carry the V.24 HDLC traffic, STUN (serial tunneling), LT2tpv3 (Layer 2 tunneling protocol), or CEoIP (circuit emulation over IP). For various reason STUN is my choice, mainly because it's well supported in old IOS releases, does not need special hardware, and is simple to get going. This is reason for the Enterprise feature set requirement as it includes support for old IBM SNA communications which was based on SDLC from which HDLC was developed.

Next time we'll look at how STUN and the router work together to actually pass V.24 HDLC traffic.

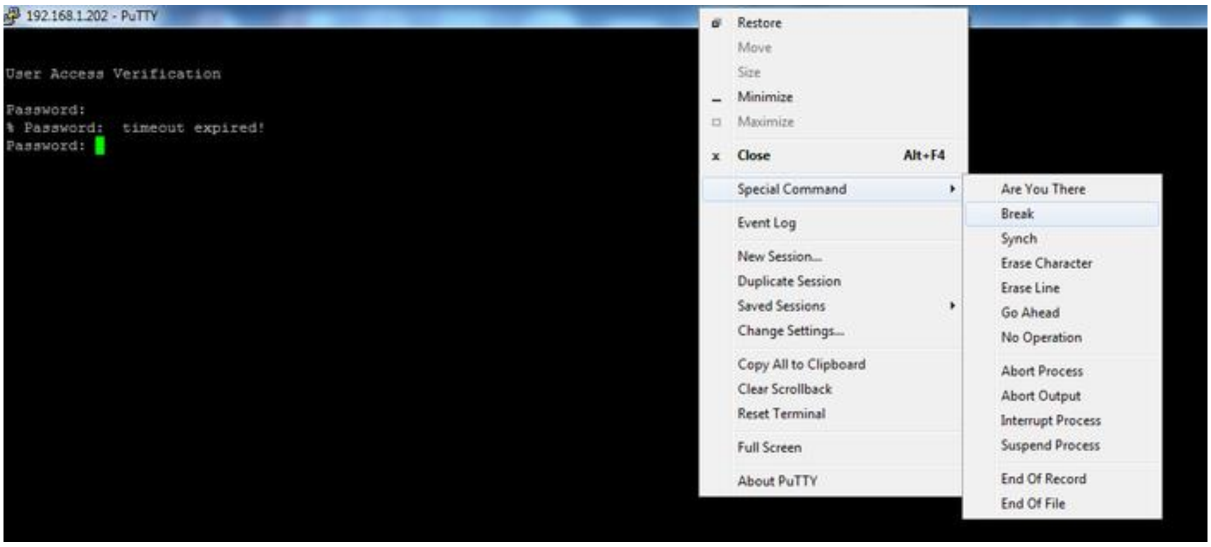
Reference 1: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Reference 2: https://www.freeccnaworkbook.com/workbooks/ccna/identifying-cisco-router-and-switch-software

A postscript to the IOS introduction, many pre-owned routers will come with a password set to protect access to the internal configuration. We need to clear the password and delete the old config. There is no point keeping the old config and it's best not to even look at it, it holds nothing of interest and may contain sensitive stuff it's best not to have. The following process bypasses the password then clears the router back to a clean factory default which also clears the password. Even if you don't have a password set, I recommend following along from the 'reset router to default' step. The IOS software is left alone, it's only the config that we're dealing with here:

Password bypass ready for reset:

Power off and power on the router. Using PuTTY, click on the top border of the PuTTY window and select 'Break', like this:



```
User Access Verification

Password:
% Password:  timeout expired!
Password:
```

The screen grab above shows the router already at the password prompt, but you need to do this within 60 seconds of power on to enter the basic rommon (ROM monitor). Type <`confreg 0x2142`> (you type what's between the <> marks) at the `rommon 1>` prompt:

```
romon 1> confreg 0x2142
```

Then type <`reset`> at the next prompt:

```
rommon 2> reset
```

The router will reboot, ignoring the previous owner's saved config. There will be various questions during the restart, just answer <`no`> until you get to the `Router>` prompt.

Reset router to default:

Now reset the router to default by following the next steps carefully:

```
router> enable
router# configure terminal
router(config)# config-register 0x2102
router(config)# end
router# write erase
```

Reload the router by typing <`reload`>. When asked save the configuration, type <no>, then confirm by typing <`yes`> or just press enter:

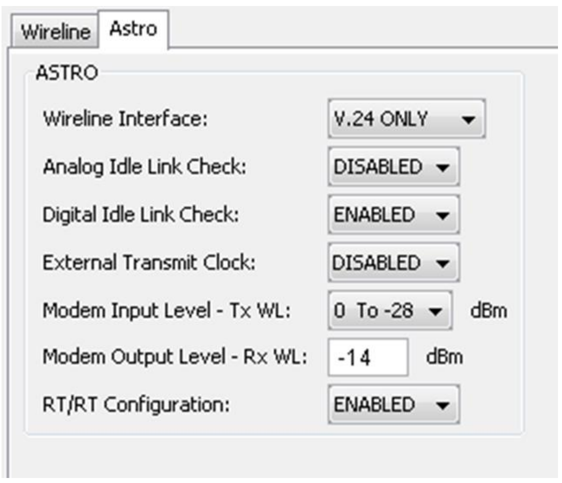
```
router# reload
System configuration has been modified. Save? [yes/no]: no
Proceed with reload? yes
```

Once the router reloads you'll get:

```
--- System Configuration Dialog ---
Would you like to enter the initial configuration dialog? [yes/no]:
```

Type <`no`> again and the router will now be cleared back to the original factory default with no password!

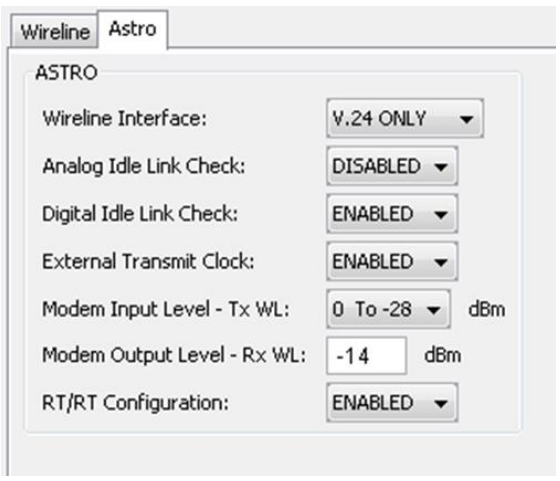
Let's get ready for the first test link. You'll need two working Quantar machines set up for P25 working with TTN4010 V.24 daughter boards fitted to CLN695X or newer WL cards. You can also use my single chip design available via the P25NX web site. Switch 1 of S101 on each V.24 daughter board should be on and the other switches off. They're often shipped all off, S1 off sets CTS off rather than feeding thru CTS from the external interface while S2 on loops RTS to CTS, it doesn't make much difference. Don't forget to put dummy loads on the transmitter outputs. You'll also need a P25 handheld. For this basic test you need to edit the code plug to enable the RT/RT config as shown:



ASTRO

Wireline Interface: V.24 ONLY
Analog Idle Link Check: DISABLED
Digital Idle Link Check: ENABLED
External Transmit Clock: DISABLED
Modem Input Level - Tx WL: 0 To -28 dBm
Modem Output Level - Rx WL: -14 dBm
RT/RT Configuration: ENABLED

Option 1 Quantar clocks TX data

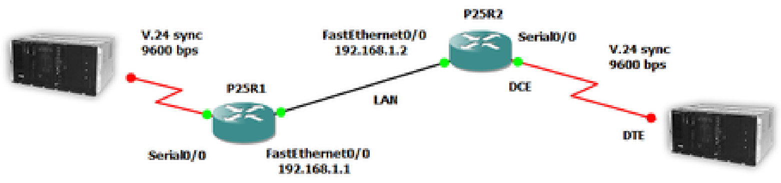


ASTRO

Wireline Interface: V.24 ONLY
Analog Idle Link Check: DISABLED
Digital Idle Link Check: ENABLED
External Transmit Clock: ENABLED
Modem Input Level - Tx WL: 0 To -28 dBm
Modem Output Level - Rx WL: -14 dBm
RT/RT Configuration: ENABLED

Option 2 Cisco clocks Quantar TX data

Now hook up everything like this, using the adapter wiring 'Option 2 Quantar TX clock in config' where the clock comes from the Cisco shown earlier this thread. You want to connect to the Quantar end of the adapter cable to port 1 on the face of the SCM:



The routers can be connected back to back with either a cross over cable or via an Ethernet switch.

Establish a console connection with PuTTY to the left hand router, and when you get to the `Router>` prompt enter the following commands:

```
router> enable
router# configure terminal
router(config)# hostname P25R1
P25R1(config)# stun peer-name 192.168.1.1
P25R1(config)# stun protocol-group 1 basic
P25R1(config)# interface FastEthernet0/0
P25R1(config-if)# description LAN
P25R1(config-if)# ip address 192.168.1.1 255.255.255.0
P25R1(config-if)# exit
P25R1(config)# interface Serial0/0
P25R1(config-if)# mtu 2104
P25R1(config-if)# no ip address
P25R1(config-if)# encapsulation stun
P25R1(config-if)# clockrate 9600
P25R1(config-if)# stun group 1
P25R1(config-if)# stun route all tcp 192.168.1.2
P25R1(config-if)# exit
P25R1(config)# exit
P25R1#
```

So what did we just do? Enable put the router into EXEC command mode, hostname names the router something more interesting than 'router', the STUN peername tells other routers who the router is, the STUN protocol group is a reference, while the word 'basic' set the STUN mode of operation.

We configure the Ethernet interface with an IP address, note how the config mode indents to take commands specific to the interface, you exit the indent with 'exit'. Next we set up the serial interface referencing the previously established STUN group, setting the speed, and telling the interface where to send the traffic (192.168.1.2 which will be the address of other router).

Check your set up by typing `show run`

You can ignore for the moment the extra default setup stuff already in the router but you should be able to see your new config statements. If all OK save it with the command:

```
P25R1# copy run start
Destination filename [startup-config]? yes
```

About half way through entering these commands the indicator lights on the Ethernet interface and the Serial port should come on.

Now we need to repeat the process on the second router with two address changes as follows:

```
router> enable
router# configure terminal
router(config)# hostname P25R2
P25R2(config)#
```

Then much as before (but I'll leave out the router prompts):

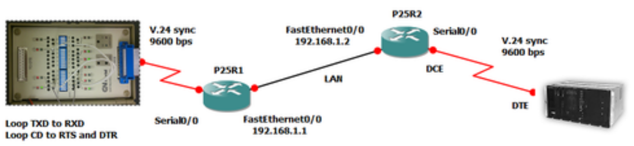
```
stun peer-name 192.168.1.2
stun protocol-group 1 basic
interface FastEthernet0/0
description LAN
ip address 192.168.1.2 255.255.255.0
exit

interface Serial0/0
mtu 2104
no ip address
encapsulation stun
clockrate 9600
stun group 1
stun route all tcp 192.168.1.1
exit
exit
```

Check your entry using show run, then do a <`copy run start`> to save. If you don't save your set up it will not last over a power down. Now for the magic (if everything is right), key up one Quantar with a handheld and the other Quantar should key as well!

Next I'll talk about how to debug this set up. I'll also explain how to test if you have just one Quantar and still want to follow along.

If you have only a single Quantar to test with then it's possible to test it against itself by looping the V.24/RS-232 connections at the far end router. You'll need to connect TXD to RXD and take the CD line from the router and use it to drive RTS and DTR back to the router. The setup will look like this:



With this arrangement every keep alive message sent by the Quantar at router P25R2 will reach the loop back (RS-232 break-out box in this case) at P25R1 and then return over the IP route back to the Quantar. Successful link up can be seen by the LED on the WL card. This flashes when the link is not up and will go steady on when keep alives are being received.

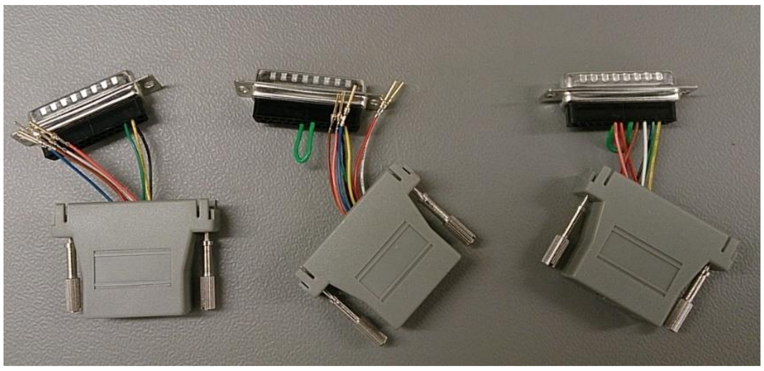
A break-out box is a very handy debug tool as it displays the states of the various V.24/RS-232 handshake and data signals. It also shows the activity on the TX and RX clock lines when the router's Serial0/0 interface is configured correctly.

You can also see a successful establishment of the STUN link by using the <`show stun`> command on the router console connection.

There is nothing like mass production to make you work on ease of assembly and find bugs.

My wife and I recently finished doing a whole bunch of RJ45 to DB25 adapters as described above using commonly available ready-made RJ45 to DB25 hardware from Norcomp (P/N RJADK25P7080831) to replace tediously soldered cables. All went well until I tested the first one and discovered that I'd overlooked documenting a reasonably essential jumper, so assembly had to be put on hold for a solution:

The Cisco serial interface (DCE) needs to see DTR (Data Terminal Ready pin 20) 'on' for operation. Since the Motorola V.24 interface on the RJ45 does not have enough pins for more than a basic complement of V.24/RS-232 handshake signals, there is no DTR to cable thru. I wasn't keen on using RTS as this would mean soldering a second pin somehow onto the existing lead. The best solution was to use Cisco's own DSR (Data Set Ready pin 6) output to drive the DTR input, linking pins 6 to 20. All I would need is some extra pins for the Norcomp connectors, so after a quick order on Digi-Key for loose pins (Norcomp P/N 100 170-101-170L001) we had the necessary jumpers under constructions. They're green in the picture below. From left to right half-finished adapters, add the jumper, finish pushing pins and then job done:



OK time to reveal the next phase!

The Cisco router is much more than just a V.24 synch to IP adapter. As may be apparent the 'baby steps' config I posted above is fine for two routers on a private LAN with static IP addresses. But how do we make this work over the Internet? What about using typical DHCP addresses? How do we protect the configuration from hackers?

The steps needed to do this require some reasonably serious Cisco-fu but conceptually it is not too hard. In essence we create a private or virtual network cloud over the Internet. This VPN (virtual private network) allows the simple config above to run protected from the vagaries of the Internet. A hub router is established which must have fixed public IP address so each remote router (using either a static or dynamic DHCP address) knows where to 'phone home'. The VPN is established using the ISAKMP (Internet Security Association and Key Management Protocol) and IPsec (Internet Protocol security) for securing our virtual LAN communications by authenticating each router to the hub. In other words, each router has a key that allows it to join the VPN and once authenticated a tunnel is created through the public Internet.

The technical details of how this is achieved will be covered in the next few posts.

As I mentioned in my last post the hub site really needs to have a fixed IP. There are ways around this but for a serious system DDNS and other tricks are a bit welfare. The upside of a fixed IP for the hub is that all the remotes can use DHCP assigned dynamic IP assignments. The routers used in my networks are 3725 models fitted with 8 port serial cards (bottom router in the stack shown in post #1).

So let’s start the hub router configuration by setting up basic housekeeping and implementing some of the hardening needed to brave the public Internet:

```
hostname P25_hub_site
enable secret <secret password for enable prompt here>
ip options drop
no ip bootp server
no cdp run
no ip http server
no ip http secure-server
ip telnet source-interface Loopback0
```

So we give the hub router a name, set a secret password for the enable prompt, drop potentially malicious packets, turn off the BOOTP service, turn off built in web services, and set telnet so the source address for establishing Telnet sessions over the VPN to remotes is the loopback address (very useful to manage your network).

We need to be able to have remote control of the hub router but let's restrict that to management from a fixed IP address and we'll implement some timeouts for good measure in case we forget to disconnect. Using a time server to set the system time is extremely useful for debugging crypto sync issues:

```
access-list 23 permit <your remote management IP address here>
access-list 23 deny any
!
line con 0
logging synchronous
transport preferred none
escape-character 3
!
line aux 0
exec-timeout 0 1
no exec
transport output none
!
line vty 0 4
access-class 23 in
exec-timeout 5 0
password 7 <secret to access router basic prompt here>
logout-warning 60
absolute-timeout 15
logging synchronous
login
escape-character 3
!
ntp server <address of a local public ntp server>
```

Next we need to establish the remote access policy for our VPN. Because we're dealing with older Cisco hardware and IOS releases, our crypto choices are a bit limited. We'll choose triple DES to protect the VPN authentication with the key shared with the remotes in advance.

When putting a router directly on the Internet you need to get serious with security. It's not so much the security of your VPN that we're necessarily concerned with, it's the risk of your system being subverted and used for some of the nasty things that happen on the Internet. You don't want to be part of that and it could be particularly embarrassing if you are sharing your employer's infrastructure (with permission of course). So do not let people shoulder surf your router configs, keep your backups encrypted, and don't give out enable passwords. For more information, take advantage of your tax dollars and review:

https://www.nsa.gov/ia/_files/routers/cisco_exec_sum.pdf

```
crypto isakmp policy 1
encr 3des
authentication pre-share
group 2
crypto isakmp key <secret authentication key> address 0.0.0.0 0.0.0.0
crypto isakmp keepalive 180 3
crypto isakmp aggressive-mode disable
crypto ipsec transform-set TS esp-null esp-md5-hmac
```

In this example we've used a group pre shared key and an address that will match the IP address of any remote. We reduce the attack footprint by disabling aggressive mode. Next an IPsec transform set is established, this is a set of security protocol settings to apply to VPN traffic. During the initial VPN security negotiation, the hub and remote need work with the specified settings. In this case we choose to encapsulate the protected VPN traffic with no crypto (esp-null) but set the authentication to be MD5 (HMAC variant). These parameters are carefully chosen for this application.

Why not use encryption for the actual VPN traffic? In some applications, such as linking over amateur radio, encryption is not generally permitted for voice or data whereas methods of secure authentication are allowed, indeed encouraged. We also need to be aware of latency; complex encryption and hashing algorithms add to the IP circuit transport delays. The relatively low speed of these older routers and the fact that most people will not have hardware crypto acceleration limit our practical choices. If this doesn't apply to you then you probably don't need my advice. More hardening is recommended, depending on your circumstances the steps listed here are a minimum. Some further useful security information is available here:

http://www.nsa.gov/ia/_files/routers/C4-040R-02.pdf

Now we need to arrange the VPN to be associated with each remote. In this next config excerpt we name the dynamic IP crypto maps P25-VPN 10 and 11, associating them with traffic from remotes 1 and 2. If you have more remotes you'd just keep creating dynamic maps for each remote. We set the SA (security association) lifetime to be 9 hours and 20 minutes. We choose the group 2 Diffie-Hellman calculation of 1024 bits and PFS (perfect forward secrecy) mode so a new DH calculation is required each time to eliminate man in the middle attacks. After this time a new SA will be established for each peer. Lastly we name the collection of dynamic crypto maps VPN1 to assign to an external IP port on the router:

```
crypto dynamic-map P25-VPN 10
set security-association lifetime seconds 33600
set transform-set TS
set pfs group2
match address VPN1-TRAFFIC
!
crypto dynamic-map P25-VPN 11
set security-association lifetime seconds 33600
set transform-set TS
set pfs group2
match address VPN2-TRAFFIC
!
crypto map VPN 1 ipsec-isakmp dynamic P25-VPN
```

Because we're creating a virtual private network, independent of the real Internet we need a virtual rather than physical address for our router, we do this with by creating a loopback interface which can have any address we like because it will never be accessible from the real world:

```
Loopback0
ip address 100.100.100.1 255.255.255.0
```

But we still need a real world connection for our hub so we next define the public interface. Because this interface is subject to attacks we need to turn off anything that's not essential, including the 'back door' Cisco management MOP (Maintenance Operations Protocol). Lastly we associate the previously defined crypto VPN map with the interface and specify the route to the Internet:

```
interface FastEthernet0/0
description Outside interface
ip address <a static public IP address> <static IP address mask>
no ip route-cache cef
no ip route-cache
no ip mroute-cache
duplex auto
speed auto
no keepalive
no mop enabled
crypto map VPN
!
ip route 0.0.0.0 0.0.0.0 <your public IP gateway>
```

Now as with the remotes we need a peer name for use by the STUN protocol, we need to define a STUN protocol group for each remote to belong to and for each physical serial interface to work with. Once done we can then configure the real serial interfaces that the remote V.24 traffic will exit on. Note that the STUN route commands specify a simple IP address that would not work in the real world, that's OK because they will only be working on a virtual network and can use any IP address we like:

```
stun peer-name 100.100.100.1
stun protocol-group 1 basic
stun protocol-group 2 basic
!
interface Serial1/0
mtu 64
no ip address
encapsulation stun
clock rate 9600
stun group 1
stun route all tcp 1.1.1.1
!
interface Serial1/2
mtu 64
no ip address
encapsulation stun
clock rate 9600
stun group 2
stun route all tcp 2.2.2.2
```

To finish the config of our hub router we need to set an ACL (access control list) so that virtual traffic knows where to go:

```
ip access-list extended VPN1-TRAFFIC
permit ip 100.100.100.0 0.0.0.255 1.1.1.0 0.0.0.255
!
ip access-list extended VPN2-TRAFFIC
permit ip 100.100.100.0 0.0.0.255 2.2.2.0 0.0.0.255
```

Next up I'll detail the changes needed to the example remote configurations of post #5 needed to work with our new hub.

OK now for the remote config. This example is for router 1, named R1P25, which can sit behind a DSL or cable router. We can use an address supplied by DHCP on the remote, the public address of the DSL or cable connection is not needed and indeed that often changes depending on your ISP. As previously mentioned a fixed IP is needed for the hub.

I've included <`comments`> for an optional fixed IP on your local LAN if you wish. You'll see that most of the sections mirror the hub router configuration:

```
no service pad
service timestamps debug datetime msec localtime show-timezone
service timestamps log datetime msec localtime show-timezone
service password-encryption
no service dhcp
!
hostname P25R1
!
enable secret <secret password for enable prompt here>
```

```
!
ip subnet-zero
no ip source-route
!
no ip domain-lookup
!
crypto isakmp policy 1
encr 3des
authentication pre-share
group 2
crypto isakmp key <secret authentication key> address <the static public IP
address of your hub router>
crypto isakmp keepalive 180 3
crypto isakmp aggressive-mode disable
!
crypto ipsec transform-set TS esp-null esp-md5-hmac
!
crypto map vpn-to-P25alpha 10 ipsec-isakmp
set peer <the static public IP address of your hub router>
set transform-set TS
set pfs group2
match address VPN
!
no call rsvp-sync
!
stun peer-name 1.1.1.1
stun protocol-group 1 basic
!
interface Loopback0
ip address 1.1.1.1 255.255.255.0
!
interface FastEthernet0/0
description Outside interface
ip address dhcp <alternatively you could specify a fixed IP i.e. 192.168.1.x
255.255.255.0>
duplex auto
speed auto
no mop enabled
crypto map vpn-to-P25alpha
!
interface Serial0/0
mtu 2104
no ip address
encapsulation stun
clockrate 9600
stun group 2
stun route all tcp 100.100.100.1
!
ip classless
<if you didn't use DHCP then add: ip route 0.0.0.0 0.0.0.0
address_of_your_gateway_router>
no ip http server
!
ip access-list extended VPN
permit ip 1.1.1.0 0.0.0.255 100.100.100.0 0.0.0.255
!
access-list 23 permit <your remote management IP address here>
access-list 23 permit 100.100.100.1
access-list 23 deny any
no cdp run
!
```

```
line con 0
exec-timeout 0 0
logging synchronous
transport preferred none
escape-character 3
line aux 0
exec-timeout 0 1
no exec
transport output none
line vty 0 4
access-class 23 in
exec-timeout 5 0
password <secret to access router basic prompt here>
logout-warning 60
absolute-timeout 15
logging synchronous
login
transport input telnet
escape-character 3
!
ntp source Loopback0
ntp server 100.100.100.1
end
```

A few notes:

Set the router's NTP client to slave from the hub router time service.

I include the Loopback address of the hub router in the ACL for remote control so that we can Telnet to remotes from the hub router without having to know their actual public IP addresses and without having to open ports on the router or Internet gateway. This is very handy if you have a bunch of remotes using DHCP working to a hub. Note the ip telnet source-interface Loopback0statement in the hub router config tells the hub that a telnet session established to a remote is to use the internal private address of the hub router rather than the external public IP address.

The `<secret authentication key>` obviously has to match the hub router's key. As this is 3DES the key will have either 24 ASCII characters or 48 hexadecimal characters.

We can check out the ISAKMP and IPSEC connections are good as follows (real IP addresses removed to protect the guilty):

```
P25_hub_site#show crypto isakmp sa
dst src state conn-id slot status
xxx.xx.xx.xxx xxx.xxx.xxx.xx QM_IDLE 463 0 ACTIVE

P25_hub_site#show crypto ipsec sa

interface: FastEthernet0/0
Crypto map tag: VPN, local addr xxx.xxx.xxx.xx

protected vrf: (none)
local ident (addr/mask/prot/port): (100.100.100.0/255.255.255.0/0/0)
remote ident (addr/mask/prot/port): (1.1.1.0/255.255.255.0/0/0)
current_peer xxx.xx.xx.xx port 500
PERMIT, flags={}
#pkts encaps: 228160, #pkts encrypt: 228160, #pkts digest: 228160
#pkts decaps: 228105, #pkts decrypt: 228105, #pkts verify: 228105
#pkts compressed: 0, #pkts decompressed: 0
#pkts not compressed: 0, #pkts compr. failed: 0
#pkts not decompressed: 0, #pkts decompress failed: 0
```

```
#send errors 1, #recv errors 1

local crypto endpt.: xxx.xxx.xxx.xx, remote crypto endpt.: xxx.xx.xx.xxx
path mtu 1500, ip mtu 1500, ip mtu idb FastEthernet0/0
current outbound spi: 0x5BE06EB9(1541435065)

inbound esp sas:
spi: 0xFF5FD812(4284471314)
transform: esp-null esp-md5-hmac ,
in use settings ={Tunnel, }
conn id: 2003, flow_id: SW:3, crypto map: VPN
sa timing: remaining key lifetime (k/sec): (4594863/148)
IV size: 0 bytes
replay detection support: Y
Status: ACTIVE

inbound ah sas:

inbound pcp sas:

outbound esp sas:
spi: 0x5BE06EB9(1541435065)
transform: esp-null esp-md5-hmac ,
in use settings ={Tunnel, }
conn id: 2001, flow_id: SW:1, crypto map: VPN
sa timing: remaining key lifetime (k/sec): (4594863/140)
IV size: 0 bytes
replay detection support: Y
Status: ACTIVE

outbound ah sas:

outbound pcp sas:
P25_hub_site#
```

You can now test the VPN connection by doing a ping from the hub router's prompt:

```
P25_hub_site#ping
Protocol [ip]:
Target IP address: 1.1.1.1
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 100.100.100.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 1.1.1.1, timeout is 2 seconds:
Packet sent with a source address of 100.100.100.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 40/40/40 ms
P25_hub_site#
```

So we've pinged the Loopback address of the remote over the VPN from the Loopback address of the hub.

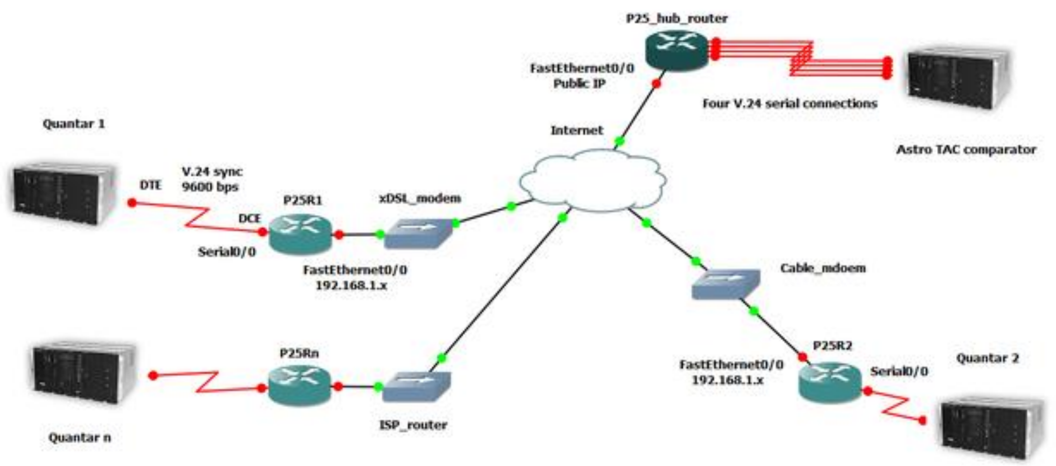
Lastly test that the STUN circuit is up:

```
P25_hub_site#show stun
This peer: 100.100.100.1

Serial1/1 (group 2 [basic])
state rx_pkts tx_pkts drops
all TCP 1.1.1.1 open 140363 218083 445205

P25_hub_site#
```

Success!



Original P25 Network

The P25_hub_router was located at a facility available to KH6MP and others

This shows a system as described above with two nodes, P25R1 and P25R2. The diagram shows an Astro-TAC comparator hooked up with four serial ports (two W/L cards). Not shown are the VPN addresses for the P25R1, P25R2, and P25_hub routers of 1.1.1.1, 2.2.2.2, and 100.100.100.1. The 192.168.1.x local LAN IP addresses are just example non-routable addresses. The ISP connections can provide a DHCP address or a fixed address as previously noted but the hub router needs to have a fixed public IP address.

I've added in another node, P25 n, to show that you can have as many remotes as you have ports available on your Astro-TAC comparator. You could also replace any one Quantar with a DIU3000 with appropriate Astro-TAC port reprograming.

Note: The Cisco symbols used in the diagram do not have their strict Cisco meaning.

Archive note: This article pre-dates P25NX, the Midwest Quantar Bridge, DVSwitch, and other networking innovations that have occurred over the last few years. It may still prove useful to those starting out with Cisco and the Quantar platform.