Layman's guide to FlexNet Drafed by N1URO June 29, 2016

Introduction:
Many folks, including Node Ops have asked me “just what is flexnet?” so I'm drafting this PDF in hopes to answer some of the more common questions I've had over the years. It's quite a fascinating system that requires very little overhead to establish longer range UHF/VHF connections.

What it is:
FlexNet is a system developed in part by a German ham Gunter Jost who also shares a U.S. Callsign... neither of which I can recall. It's been called everything you can possibly think of but after doing some deep debugging I have found it to be the following:

1) a complex ax.25 router
2) an ax.25 proxy server
3) a high end digipeater
4) a smart passthrough
5) an intelligent encapsulator

What it is NOT:
While FlexNet's motto is “we route EVERYTHING”, as those of us familiar with IP and NetRom know, to truly route any protocol there must be a matching route table for it to know just where and how to route that particular protocol. This is what had our EastNet Ipers baffled because it had NO IP route table included yet it supports axip and axudp protocols on an ethernet based system. So, giving some leeway as to their claims it routes everything a more accurate way of describing this system is that it encapsulates everything but in a very unique way.

How it works:
FlexNet as some have noticed is a chatty system. Nodes are known as Destinations or Destis for short. Each desti links to it's partner using a links table. FlexNet then shares two key elements with all of it's neighbors:

1) round trip times
2) routing information

Approximately every 3 minutes, FlexNet will send a packet to it's neighbors to which the neighbor destinations will then send a reply packet back to the originating station. It then calculates the round trip time for display and routing purposes. Since it also is a real-time mode router when the round trip time (rttl) changes routing information is almost immediately sent out to its neighbors to reflect this new calculation. This is key where you have a redundant path from point A to point B as FlexNet will route you to point B using the fastest path it sees available. This is all natively handled at the ax.25 layer as this is the protocol we're mainly forced to use on UHF/VHF packet networks.

The other part of it's transmissions is that it shares its dynamic routing tables with its neighbors with a detrated rttl per desti based on the rttl it receives from its neighbor. Once the initial table is filled, then mainly all that is sent out are those destis who's rttl has changed. Again, in the event of redundant paths this is key for FlexNet to know which is the quicker path. Since at the time of its development there were no ax.25 packet identifiers (PIDs) available, the author chose to use PID CC (RARP) as that was seen to be the least used of the PIDs allocated for ax.25.

At first this sounded like a major setback for those who enjoy IP on RF but it's really not necessary. Both FlexNet's “pings” and route information use binary under a standard text-based ax.25 protocol layer. More on why later.

The Need For Speed:
FlexNet also does what no other TNC based digipeater does in regards to digipeating frames... true hop to hop acknowledgment on the frames, but with a slight twist as well. We all know ax.25 is a 16-bit protocol that can use numbered or unnumbered type frames. Typically in such entities as beacons, they're sent unnumbered as they're not in need of any error correction. In numbered frames, these type are seen as more important frames in need of error correction to insure that any stream is held in tact. This error correction is in the form of an 8-bit system numbering frames 0 through 7 and when 7 is reached it resets back to 0. When something such as PBBS mail forwarding occurs, a stream of frames are sent in a numbered sequence. FlexNet out performs NetRom as it will only acknowledge the first and last frame within this 8-bit sequence whereas NetRom must acknowledge each individual frame sent. Also NetRom adds an additional 20 bytes to the protocol header thus making the data segment of the total 256 bytes lower (to 236 bytes). If you add IP under NetRom your taking away even *more data space* in each frame. Since 5 frames may pass without the need of an acknowledgment, FlexNet at 1200 baud may seem almost as fast as a NetRom link at 9600 baud.

If a packet in the stream is missed by the remote however, the remote will request a refeed at the numeric it failed to receive thus not breaking the error correction within the ax.25 layer. This is also very handy for ascii uploads by a remote user to the OSI Layer 7 (application) such as a PBBS. When a refeed is requested, the stream resets at the numeric it needed to refeed.

We Route EVERYTHING:
As I mentioned earlier, FlexNet's motto isn't quite 100% accurate and confuses many Node Ops, especially those familiar with the various flavors of xNOS. When FlexNet was deployed (in a rapid deployment) the goal was to rid the network of ANY other protocol except vanilla ax.25... and for approximately 5 years they were successful in doing so until I found the key secret and was able to draft and test a design that worked flawlessly. The secret in passing IP through FlexNet from point A to point B is to create and engage an arp entry to the remote node you wish to pass IP with AND to install a proper path exactly reverse from that remote site to your end siting *only the ssids* of your direct FlexNet neighbors at both ends. What this does is creates a virtual circuit so in IP you wish to set your ax.25 stack to Virtual Circuit (VC) mode. Once you have successfully created your virtual circuit you may begin exchanging anything you wish to route with your remote partner. In this example, I'll use N4LEM as a host and send a ping to W4OT which is 5 hops away with all the traffic encapsulating under ax.25 through FlexNet...
First the arp entry:

```
                44.98.16.2 ax25    W4MLB-13          CM                ax1
```

Now the route entry:

```
             44.98.16.4 via 44.98.16.2 dev ax1  src 44.98.16.1
```

Ok all looks good so far as the path to W4OT goes to W4MLB. Now let's ping it:

```
ax1: fm N4LEM-12 to W4MLB-13 via N4LEM-2 W4MLB-3 ctl I37^ pid=CC(IP) len 84
IP: len 84 44.98.16.1->44.98.16.4 ihl 20 ttl 64 DF prot ICMP
ICMP: type Echo Request id 19851 seq 0
0000  ......tW�_.............................................

ax1: fm W4MLB-13 to N4LEM-12 via W4MLB-3* N4LEM-2* ctl I04^ pid=CC(IP) len 84
IP: len 84 44.98.16.4->44.98.16.1 ihl 20 ttl 63 prot ICMP
ICMP: type Echo Reply id 19851 seq 0
0000  ......tW�_.............................................
```

Here you see the digi paths from the direct neighbor FlexNet nodes but at the OSI Layer 7 it's much easier to read:

```
n1uro@n4lem.ampr.org:/uronode$ ping w4ot
ICMP Echo request sent to: 44.98.16.4
ICMP Echo reply received from: 44.98.16.4
Ping completed in: 2106ms (ttl=63)
n1uro@n4lem.ampr.org:/uronode$
```

If there is a redundant path available and the primary path fails even during an IP session, the virtual circuit *will recreate itself and **not** drop the established IP socket*! In watching sniffs, I've seen SMTP sessions where N2NSA (MFNOS) was sending me an email to N1URO and the 9600 baud path up through WB2ZII and K2PUT failed, however a redundant path via W1HAD was there to reroute the ax.25 layer and the virtual circuit recreated while the IP socket transporting the SMTP data remained in tact 100%. This makes FlexNet act to the ax.25 protocol encapsulating IP parallel to the global wired internet systems using BGP (border gateway protocol) to exchange routing information amongst Internet Service Providers.

NetRom may be handled too if both ends have the ability to digi NetRom OBS between each other. As with IP, NetRom uses a virtual circuit to stream it's data through. As of this writing I know Xnet has this ability as I've tested it and I may be mistaken but I think X1J-4 may as well. Another way of routing NetRom is under IP using axip. Since IP has already created a virtual circuit within FlexNet, axip easily delivers NetRom frames through and very reliably.

Rose transports parallel to NetRom for inter-connections with nodes such as FPAC, F6FBB, and others.

Outreach:
Since FlexNet uses rttl and not a preconfigured mathematical equation to derate nodes per hop, this means that passing IP and/or NetRom from point A to point B has a much further outreach. No longer are we limited to the 6-8 hops of NetRom and the old specifications. With FlexNet, the rule of thumb is that if you can see it in the Destis tables, you should be able to connect to it. After all it's “real time”. This means even if a node may be 14 hops away, you should still be able to create and maintain a virtual circuit for IP or a digi path for NetRom, and if you're using IP, the amount of apps you can service end users is almost endless. At one point EastNet hosted several web servers on RF that worked just fine. You can host FTP servers, converse, databases, the sky is almost the limit. That part of the equation is up to the Node Op.

More Error Correction:
The FlexNet team developed a new type of eprom we now use called 6-pack. Unlike standard KISS, 6-pack adds an additional bit to the communications between the PC and TNC, which allows FlexNet to transmit slightly faster than your standard KISS environment. For those who wish to be creative, you may also configure multiple TNCs using a token ring configuration and strap them together. Also 6-pack delivers additional error correction on the serial interface between the PC and TNC. While FlexNet may use KISS to speak to TNCs, 6-pack is the faster and more preferred method. Native Linux supports 6-pack with the -6 switch in mkiss.

Summary:
As you can see, there's many more bonuses with using native FlexNet than there are with standard NetRom or even classic IP configurations. Your coverage is greater, your speeds are enhanced, and your error correction is much better. While there's a bit of a learning curve, the EastNet email list is there to help.