FlexNet

THE EUROPEAN SOLUTION

Gunter Jost, DK7WJ
Lichtenbergstr. 77
D-6100 Damstadt
DK7WJ @ DB0GV.DL.EU

Joachim Sonnabend
Annastr. 4
D-6082 Moerfelden
DG3FBL @ DB0GV.DL.EU

FlexNet-Group
Bunsenstr. 2a
D-6100 Damstadt
Fax: 49 6151 895718

This paper describes the design and implementation of FlexNet. This is a packet switch software with a complete new design made in Germany. It is beginning to become the standard in Central Europe. Running on a dedicated hardware with an unique design it is easy to put cheap and reliable packet switches on our mountain sites.

1. Introduction

In the last years of increasing Packet Radio activities, some groups in Germany began to develop hard- and software for network nodes. In the Frankfurt area the RMNC (Rhein Main Network Controller) was developed by a small group of people. This is a nodecomputer, optimized for high data thorughput and an extreme good price/performance relation. Parallel to the hardware, an absolute new packet switch software was developed in Darmstadt. After only 15 months of distribution, this software is running on more than 100 RMNCs in DL and the neighbour countries. A portation of the system to other hardwares is planned. RMNC/FlexNet is one of most modern, fastest and on the same hand cheapest nodecomputer concept available in Europe. A lot of nodes change to the RMNC/FlexNet, because their old packet switches have reached the barriers of their efficiency or an extension would be too expensive. On the other hand there is a built-in autorouter in the newest version of FlexNet, which is completely new developed and has been proofed to be extremely efficient.

2. Network Topography in DL

For the understanding of the FlexNet layout we want to explain some aspects of the PR situation in DL. The frequency assignments and the goverment regulations had a lot of influence on the development of our PR network. Network nodes and mailboxes which are operated unmanned need a special license. This has a lot of disadvantages which we don't want to discuss in here. But there is a kind of coordination which is an

advantage, because we realized a relative good structurized network. Fortunately there were no restrictions in using new protocols. By this reason it was possible to create several interesting projects for digipeaters and network nodes. Today there are more than 100 "official" nodes in our small country. Because of the frequency situation most of them only have one user channel (mostly on 70cm). The nodes are connected with exclusive links on 23cm. This is the reason why these lines are free of collision, undisturbed and why it is possible to reach a high performance with a relative low datarate. Lots of lines are still working with 1200 Baud half duplex, but mainly the most frequent lines are changed to 9600 Baud full duplex. Because of the-limited frequency assignments, higher data rates are hard to realize. This is the reason why we are thinking about changing to the microwave bands. The links have a length of 80 - 200 km and the nodes have about 2-6 links to othes nodes, each on an exclusive frequency pair. For this a huge effort of RF components is necessary. This explains once more the call for a cheap nodecomputer with great performance data.

3. Hardware: RMNC

The RMNC is a modular system. All boards have eurocard format (100 * 160 mm). They are connected with a 64 pin backplane for parallel data exchange. To make it cheaper, all components which are common for all channel processors are seperated on a so called reset-10 board. This board has a reset generator with watchdog, a clock generator and 32 switch IO's for remote controls. The channel processor has already been developed in 1986 and contains the following components:

- CPU MC6809 with 4 or 8 MHz clock
- VIA 6522 for buscomunication
- SCC 8530 for HDLC-IO
- 27256 EPROM 32kByte
- 2 * 43256 RAM 64kByte, one with battery backup for parameter storage
- TCM 3105 Modem for 1200 Baud, optional
- Modem disconnect plug for external modem

The boards are connected with a backplane which is availible in the surplus trade as an european industrial standard. The whole computer is supplied with 5 volts only. The boards are distributed as kits. However, the board layouts are in the public domain for noncommercial use. If you are able to get the-components quite cheap, the price for the whole RMNC including backplane, case, power supply and 6 channels could be under $350.

4. Software: FlexNet

FlexNet was developed out of classical digipeater concepts. The first RMNC-Software (Ver. 1.x) had a list of its neighbour digipeaters which was coded in EPROM. So every digipeated frame has been sent to the right channel. This made it possible to manage exclusive links. The aim for the development of the first FlexNet version was to get the connectability of the node and a Hop-To-Hop-Acknowledge with simulated digipeating. with this

concept the disadvantages of the Net/Ram-concept, which spred around in DL, should be avoided.

FlexNet was developed without third party source code. The basics for the development was the AX.25 protocol as well as monthlong brainstorming with friends. Thanks to the programming language C it is possible to port the program to other computers. Because of the bad performance and the inefficient code we did not port it to the **Z80-TNCs.**

The first **FlexNet** implementation on an RMNC **came** in the beginning of 1989 with the version number 2.0 to make a distinction to the old RMNC software. The 2.x version and the new version 3 have the following features:

- Connectability, the user gets a Ctext
- Callable infotext (Help, Info, News)
- All parameters, **activity-** and link informations are readable
- Conversation mode for roundtable discussions
- Complete remotecontrol for the Sysops (links, baud-rates, parameters, infos, beacons), all commands in plain text
- There is only one **callsign** for the whole node, made by the master-slave-principle. The routing to the different channels is made by analyzing the next **callsign** (linklist) or by SSID, if there is more than one user channel
- Users cannot access so-defined link channels directly, therefore the traffic on the links is collision free
- Simulated digipeating with the known scheme
  "Connect ‹User› via ‹Node› [,‹Node›]"
  UA is delayed until the connection over the network is made
- Flow Control with **RNR/RR** independant for each hop
- Window size for QSO is equal to the sum of the maxframes of the involved nodes, as a result good throughput on long range connections
- Selective Flow Control for each QSO made possible by virtual circuits, so congestion cannot occur for fast links when slow links on the same node have problems to get rid of their frames
- Round-Trip-Timer replaces FRACK parameter
- Errormessage on breakdown of the connection:
  "<Node>: Link failure"
- No timeout for **QSO's** via the node

There were a lot of improvments made on **maintanance.** On general demand we built in a connect command. This is done without changing the **SSIDs.** Behind the node processing the C-command, the same path will be created like it would have been with normal digipeating.

Example: ‹**User**› to <Node>: "**C ‹Friend›**"
The node works to the other side with the callfield
<User> **-›** <Friend> via <Node>*

With this trick our friend always can connect back to us without the need for us to tell him the path to use.

5. FlexNet Autorouter (FlexNet V.3)

The autorouter of the FlexNet V.3 is a complete new development. Because of the bad performance of the known routers we decided not to make it compatible with them. Wo did not like the long node lists where only a few nodes where really accessible. And we wanted to build a router which is able to adapt himself quickly to changes in topograhy and failed or overloaded links. And of course it should measure the actual availibility and performace of the links so that it is really able to find the best way to any known destination. The development started in the beginning of 1989. The first tests with several nodes were made in winter 89/90. In August 1990 the router was released with FlexNet v.3.

The linklist (list of the neighbours) will be put in manually by the sysops. (An automatic detection of neighbours by using broadcasts would be possible, but we don't think its useful. In Germany we don't need it because of our fixed link concept).

Internode communication only takes place between neighbours who both have their partner in the link lists. By this unwanted nodes can easlily be locked out.

There is a permanent L2-connection between the nodes, by this we can exchange destination infos and are able to test the functionality of the links. So it is easy to detect a breakdown of a link or of a neighbour and the router is able to react immediately.

Every 5 minutes FlexNet makes a runtime measurement for a 200 Byte testframe within the internode connection. With the last 16 measurements the average runtime to the neighbour will be calculated. With this calculation the line utilisation and the influence of retries will be recorded. This time is used for routing and the users will be informed about it in the linkinfo. The best way to a destination is the route with the smallest actual runtime calculated by adding the link runtimes.

Links to neighbours which cannot handle the internode protocol1 (i.e. mailboxes) are checked with periodical connect tries. This also makes it possible to calculate the runtime and to detect a breakdown after some tries without success. An UA or DM is sufficient as a response, so mailboxes should lock out the call of their local node.

The so-collected destination infos are exchanged with all FlexNet neighbours. Because of the internode connection no periodical repetition is neccesary. When the runtime to a particular destination changes significantly, the info is updated. By this the network can quickly react to any changes. When a link breaks, the internode qso goes down after a few minutes. Then all routes via that link are deleted immediately.

On each node the user can ask for the runtime to each known destination, and if he is interested in how the router works, he can ask for the complete path to a destination. To get this

information the node needs some special internode traffic because each node only knows the next downstream neighbour. Therefore the path info may be delayed for some seconds.

By the use of an adaptive hold down timer bad destination news are growing faster than good ones. Because unreachable destinations and unusable links are quickly detected, the destinations shown in the lists were really available a few minutes ago.

Sink-Tree-Algorithm with adaptive Hold-down-timing guarantees loop free routes. There is no necessarity for a Time-to-Live-Counter.

Virtual Circuit: There is no necessity of a Layer 4 with End-to-End-Acknowledge. One step connection all over the whole network is possible, the user doesn't have to adapt himself. There is no need for an 3-step up/cross/downlink setup, however users can do that if they like it. Communication establishment with C-commands or one step with the command:

"Connect <Friend> via <first node›,‹last node>"

or

"Connect <Node> via <first node>"

The path is allways reversible because the first node is included in every frame. The connected user is able to recognize the way back, even if the QSO was build up with C-command and some detours. He always sees a frame with the first and last node in the digipeater callfield.

Because there is no L4 it is not necessary to use any fragmentation on long I-frames. The whole routing is done in the AX.25-digipeating field. We got it with AX.25, and now it is very sensefull to use it for routing.

At every node on the way to the destination it is possible to get the data of the QSO's and with these data (poll or reject states, unacked frame counts) it is easy to find bottlenecks in the network (and it's fun to look around what's happening). The overhead by transmitting the entrance and exitnodes in the digipeater field is not larger than at existing datagram protocolls (L3/L4-header).

It is im portant for other protocols and experiments that FlexNet routes everything: Frames that do not belong to a running AX.25 QSO are routed as datagrams (i.e. UI's), when an SABM is seen, a virtual circuit is built automatically. In both modes the PID's will be passed through unchanged. So all modes including TCP/IP can use FlexNet and draw advantage from the autorouter.

6. Performance of the actual RMNC implementation

Today maximum 9600 Baud fullduplex/channel, extension to ≈64kBaud possible and planned
16 channels/node
More than 100 QSO's/channel, ≈200 QSO's/node.
Maximum data transfer rate of the whole node is ≈500kBit/s
570 destinations storable
20 links (neighbours)

The Master EPROM with the parameters is generated by a supplied parameter compiler running under MS-DOS. The input is a text file with the same syntax used for remote control. No patches are neccesary. Slave processors allways have the same code, parms are downloaded from the master processor after reset.

7. Future development

With the 6809 we have reached the limit of the FlexNet V.3 by means of code length and data space. Because of the low costs for a FlexNet node we didn't think about a larger computer until now. FlexNet's reputie is to be very fast and efficient so we don't think about porting it onto slower computers with lower data throughput. Nevertheless it is very easy to port the software to any other computer because allmost the whole code is written in ANSI C.

In the meantime we are discussing about porting the code to PCs with plug-in HDLC boards, however that might be a more expensive solution without performance gain.

Therefore we are developing a new channel processor for the RMNC, which will have more space for code and data and which will be able to handle faster links. It should be integrated into the existing RMNC's to save the investments already made.

The technical data:
- Processor MC 68302
- 1-4 MB RAM, 512 kB EPROM
- DMA-transfers to the parallel backplane with ≈8MBit/s
- 2 channels/board with max. 1MBit/s
- RS232-port

On that board we can run TCP/IP, S&F functions and lots of other things not yet in mind. For the high level functions it is only needed once in the system as a master processor, however for fast links it can also be used as a slave. Then the data rate on the backplane will draw advantege of the DMA.

8. Conclusion

With RMNC/FlexNet a Packet Switch was created which is optimally adapted to the german network structure. It was lot of work to build a completely new design, but now we know that it made sense. We did it for fun, and we had lot of fun. Especially we feel that it is more satisfying to find self made bugs than scratching around in sources from other people.

Already without the autorouter the users liked FlexNet because the design and handling is easy to understand. The network is fully transparent for other protocols. Now with V.3, users and even sysops no more have the need to look at the network topographie. There is not one parameter for the router sysops can play with, all informations except the neighbour list is automatically built. So users can say "C <mailbox> via <node>" to their TNC and it will work on the best route available. It is nearly impossible to find a better route to the wanted destination than the router chooses. Sysops like FlexNet because they need no tables with parameter lists, everything can be controlled with plain text commands. Maintenance of the links is easy, the node keeps track with changing link qualities. There is no need for unproto tests to check the links. Even worse, it is senseless to check a link with unprotos like "‹sysop› to <test> via ‹mynode›, ‹othernode›, ‹mynode›", because this frame might be routed through some neccesary deviations and come back with the original callfield.

All frames running through the network carry the information about the originator and the first node on which it came into the network. Control authorities think that to be very useful.

In the meantime foreign groups are interested in FlexNet and wo hope to be able to support them. RMNCs are now running in the GDR, Belgium, France, Switzerland, Austria, Italy and Hungary. Especially in eastern countries our cheap concept is of great interest.

9. Acknowledgements

We want to thank the users and sysops in our area who suffered from numerous bugs in the preliminary versions of FlexNet. Sysops have changed EPROMS many times and even had to change the K-sockets because they were worn out. Lots of detailed bug reports helped to make the code as stable as it is today. We also want to thank the software gurus wo put their ideas to the public domain. The brainstormings during many meetings brought lots of ideas which we only had to realize. We just tried to make it better, but we had an easy job by starting quite late and looking at other concepts.

At last thanks to you for reading this paper although the english might not be very professional. In the near future we will have an english documentation written by an american ham.

Reference: Tanenbaum, A.S. Computer Networks (2nd edition), Prentice Hall: Englewood Cliffs, NJ. 1989