Packet White Paper: 97 Paula Dowie
August 2000

NET/ROM CIRCUIT RESET

STATUS OF THIS MEMO

This memo provides information for the Amateur Packet Radio community. It proposes an extension to the NET/ROM protocol, in order to speed up the detection and tear down of failed circuits. Distribution of this memo is unlimited.

TABLE OF CONTENTS

1. INTRODUCTION

The Amateur Packet Radio (aka "Packet") network consists of a loosely connected mesh of "nodes". Originally these nodes ran "KA-Node" firmware in Terminal Node Controllers (TNCs). These nodes operated only at layers 2 and 7 of the ISO 7-layer model.

Over time, the KA-Node firmware was supplanted by the superior NET/ROM firmware, presumably so called because it was network software in a ROM (Read-Only Memory). These nodes operate at layers 2, 3, 4 and 7 of the ISO 7-layer model.

NET/ROM nodes make periodic routing information broadcasts, which allow them to discover each other and automatically build a network.

When an end user makes a connection through the network to a distant user or node, the nodes along the "route" establish AX25 layer 2 (L2) "links" between themselves. These links carry the NET/ROM protocol frames, which occupy ISO layers 3 (L3) and 4 (L4).

The L3 portion of the NET/ROM frame contains the source and destination callsigns and a "Time To Live" (TTL) counter. The L4 portion carries information used to set up, maintain, and tear down "circuits", and to transfer data on those circuits. One type of L4 frame carries the data payload.

Data frames sent on a circuit are acknowledged by the receiving end. If no acknowledgement is received within the "L4 Timeout" interval, the data is sent again. This happens up to "L4 Retries" times, after which the circuit is torn down. Receipt of an IACK (Information ACKnowledgement) frame resets the retry counter.

Nowadays NET/ROM firmware has largely been superseded by software running the NET/ROM protocol on computers.

2. MOTIVATION

If a Packet Radio network node crashes or is rebooted, both of which are fairly common occurrences in an amateur network, all NET/ROM L4 circuits originating or terminating at that node are lost.

This may leave the other end of each failed link re-sending frames every "L4 timeout" interval (usually 2 minutes). A dead node cannot receive these frames, and a rebooted one simply ignores them.

This is not only a waste of time and bandwidth, but also increases the likelihood that a circuit number becomes reallocated. It also leaves applications "hanging", waiting to deliver data that can never be delivered.

Eventually the sender's L4 retry limit (usually 3) will be exceeded, and up to 3 DREQ (Disconnect REQuest) frames, spaced 2 minutes apart, will be sent before the sender finally gives up and informs the application or user that the circuit has died. This process can take up to 12 minutes, which is unacceptable from a user's point of view.

However, if the still-alive end of a "dead" link is NOT sending data, it cannot detect the broken link. All it can do is time out after a period of inactivity.

Inactivity timers are typically set to 15 minutes, but some links, e.g. those used for infrequent CHAT traffic may have no timeouts.

Because there is no L4 equivalent of the L2 T3 (Link check) timer, "zombie" links without application layer timeouts can persist for a very long time, wasting resources.

3. DISCUSSION

NET/ROM L4 doesn't have the equivalent of AX25's <DM>, or TCP's RESET mechanism. Apart from connection request (CREQ) frames, it simply ignores frames with an unknown circuit number. For reasons outlined above, this was a bad decision by the NET/ROM protocol designers.

It is suggested that, instead of ignoring "obsolete" frames, i.e. those with unrecognised circuit numbers, they should be actively rejected. The problem is, how to achieve that without breaking the existing protocol?

Several options were considered, all of which had the potential either to break existing NET/ROM implementations, or to require complicated modifications to the protocol.

A major problem is that most of the existing NET/ROM L4 frame types that can be sent from a node must include the other end's circuit number. But after a crash, the node no longer knows that number.

Every NET/ROM L4 "circuit" between a pair of nodes is identified by a pair of numbers. For each node, "MyCct" is the local circuit identifier, and "YourCct" is the other end's circuit identifier.

MyCct is unique on any node, it is used to find the circuit to which incoming frames belong.

YourCct may not be unique, because a node might have several circuits to other nodes whose "MyCct" numbers happen to be the same.

The normal state of a circuit is represented below:

```
   Node A                    Node B
-------------             -------------
MyCct YourCct             MyCct YourCct
-------------             -------------
  22    351 ---------351---> 351   22
  22    351 <--22----------- 351   22
```

When node A sends traffic to node B it includes only the local circuit number of node B (351) in the frames. When Node B sends back to node A, it uses only the local circuit number of node A (22).

If node B is rebooted, the failure state looks like this:

```
   Node A                    Node B
-------------             -------------
MyCct YourCct             MyCct YourCct
-------------             -------------
  22    351 ---------351---> 0     0
```

Node A sends a frame to node B using circuit number 351, but node B has no record of that number. But neither does it know node A's MyCct number (22), so it can't send anything back.

Use of the DREQ frame with combinations of CHOKE and NAK flags might have been a possible solution, but had too much potential to break existing implementations.

In the end it was decided that the simplest solution was to add a new NET/ROM L4 frame type. Existing software that properly follows the "Robustness Principle" should not be affected by the new frame type.

4. THE PROPOSAL

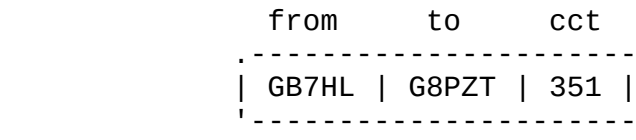
The proposed solution is to add a RESET frame type to the NET/ROM transport layer. The suggested "opcode" is 7.

If a node receives a frame for an unknown "MyCct", it should return a RESET frame containing the same number.

The combination of MyCircuit and MyCallsign is unique at any node, thus the combination of TheirCircuit and TheirCallsign is also unique. As the returned frame originates from TheirCallsign and contains TheirCct, the local circuit can be located and destroyed.

For example...

Having been rebooted, G8PZT receives this NetRom frame from GB7HL, relating to a previous connection...



Circuit 351 doesn't exist at G8PZT, so it sends back the following frame:

When GB7HL receives this frame, it searches its L4 circuit table for a match with TheirCct=351 AND TheirNode=G8PZT. If it finds such a circuit, it can kill it.

Received L4RESET frames which do not match any circuit must be ignored. A L4RESET must not be sent in response to an L4RESET.

If a node receives a frame, other than a CREQ or L4RESET, IACK or DACK, for which there is no existing circuit, it should return a L4RESET to the sender.

The received frame could be CACK, INFO, IACK, DREQ, or DACK. Let's assume we have just rebooted and receive one of these frames carrying an unknown circuit number:

CACK: The other end thinks it is connected to us, and will try to send us data, or expect data from us, yet it's a dead connection. The user could be left hanging for 6 minutes or more, so it's important to send a L4RESET for this one.

INFO: The other end expects a an IACK response, and will keep trying until the (L4retry interval * 3) expires. Again, the user at that end is left hanging on a broken connection, so we MUST send a L4RESET.

IACK: These are acks of our previously sent data. We may have sent a bunch of INFO frames, followed by a DREQ while the IACKs were still in the pipeline. By the time they arrive, our connection will be no more. In this case we MUST NOT send L4RESET frames because they would be superfluous. Both ends know they have disconnected.

DREQ: The other end thinks it is connected to us and is trying to disconnect. It will try for (L4timeout * L4retries), which is usually 6 minutes. During that time, resources are being hogged unnecessarily, and the user might be waiting for confirmation of the disconnect, We MUST send L4RESET in this case.

DACK: The other end is acknowledging our DREQ, so both ends already know that the connection has ended. There is no point sending a L4RESET in this case.

In most cases a reset situation would occur as a result of one end forgetting about an existing circuit, e.g. after a reboot. But in some cases it could happen during normal operation due to excessive delays in the network.

NB: By itself, the proposed circuit reset cannot detect broken links in the "receive only" case, i.e. where all the traffic was flowing FROM a node which subsequently fails. But combined with an application layer "keep-alive" or timeout, it can greatly speed up the detection and tear-down.

5. AUTHOR'S ADDRESS

Paula Dowie
Email: g8pzt @ blueyonder.co.uk
Packet: g8pzt @ gb7pzt.#24.gbr.eu