M.I.T. Laboratory for Computer Science August 4,1980

Computer Systems Research Division Request for Comments No. 191

High Level Maintenance Tools for Local Networks

by
Craig R. Leckband

Abstract

This paper addresses local network maintenance from a systematic viewpoint. Previous attempts at providing network maintenance tools have lacked focus and scope. These efforts have been too low level. This paper introduces a design of a network maintenance center. There would be a center responsible for each subnetwork. There are two logical components to this center. There is a component consisting of data collection and status request tools. The second component is a high level interface to the user that interprets the mass of low level data that is collected.

---

1. Overview

Isolated tools have been developed to aid in maintaining local networks. Because each of these tools was developed in an ad hoc manner, they offer a potpourri of services. The existing tools suffer two major drawbacks. The maintenance tools offer low level assistance. Moreover, they do not offer a consistent, unified approach to network maintenance.

What I am doing to alleviate this situation is to devise network maintenance tools in a systematic top-down manner. The problem has been broken down into three separate issues:

> What high level commands should be provided in the set of maintenance tools?
>
> What existing tools are useful to this end, what tools are necessary to fill in the holes, and how are they linked into a unified tool?
>
> How is the necessary information obtained (what changes in protocol are necessary)?

2. Approach

The set of network maintenance tools would appear to be one tool to the user. These user visible tools would be the network maintenance center(NMC). The reason for having only one set of maintenance tools visible to the user is to promote ease of use by providing a uniform user interface. Furthermore, this scheme affords the user a high level language for network maintenance. The NMC would integrate several support tools together. In this manner the NMC would be able to sieve through the information provided by these support tools in order to respond to high level requests made by the user.

The NMC, then, consists of two logical components. The front-end provides the user interface to the NMC by offering the user a high level command language and interpreting the low-level information provided by the maintenance support tools. The back-end consists of the set of support tools that collect information from the network.

The NMC is local in nature. There would be a NMC for each subnet in the network. Each NMC would be responsible for its own subnet, but would try to answer inquiries about other subnets. Some inquiries would require information gotten from other subnets in the system. For example, the question 'Why can't A talk to B?' would require non-local information if A and/or B are not local to the NMC where the question was raised. Also, it may be useful to generalize the set of user inquiries so that a user could find out status information from non-local subnets. The manner in which non-local information would be obtained is to have the local NMC connect itself to the NMC to be queried via a high-level end-to-end protocol (such as TCP) and have the local NMC question the other by using the user command language. The user has the option of connecting directly to the foreign NMC and obtaining information about a non-local subnet in that manner.

The NMC's might be arranged hierarchically. This would help insure that some central body would coordinate repairs. One could imagine that a NMC would be required to report to a central authority (or even a "regional" authority) when it detects a failure. Presumably the authority would be responsible for taking corrective action.

2.1 Front-End

The most important function of the front-end is to process and respond to user queries. The front-end includes a parser for the command language and a set of action routines. There is an action routine that corresponds to each input command. An action routine is responsible for collecting and digesting the information necessary to respond to a user query. This information is gained by querying appropriate databases administered by one or more back-end tools and by obtaining information from the network directly by commanding an appropriate back-end tool to do so.

It may be useful to have the front-end continuously trouble-shooting the local subnet in addition to processing user queries. In this manner the NMC could serve as an alarm system for the local subnet. The front-end would scan incoming status information and watch for patterns of network behavior that indicate trouble. Examples of such patterns would be frequent gateway restarts, frequent loss of token (LCSNET), retransmission of many packets, or network saturation. Actions taken in response to finding a network problem would vary according to the severity of the problem. The weakest response would be to have the NMC report the problem in a trouble log. A serious network problem might require the NMC to report the problem directly to one or more of the local subnet maintainers.

The syntax of the command language has not yet been determined. The syntactic complexity of the command language will be kept to a minimum while providing a robust interface to the user. A more important concern is what types of questions will be supported in this command language. The following is not intended to be an inclusive list:

> Why can't host A talk to host B?
>
> Why is the network so slow?
>
> When was the last time the network failed (lost the token)?
>
> When was the last time that each (local) gateway was rebooted?
>
> Are there any discrepancies between the connectivity of the network and what each (local) gateway perceives as the network connectivity?

Are there any discrepancies in the aggregate number of packets sent, received, undeliverable, and refused?

What is the most active link?

What is the most trouble-some gateway?

What hosts (links) are down?

What is the recent history of hosts that have failed and their respective downtimes?

What is the recent history of gateway failures?

What is the recent history of LNI failures?

What is the status of other subnets? gateways?

In addition to the above queries, the NMC may allow maintenance personnel to initiate some actions. Examples might be downloading diagnostics onto gateways. It may also be desirable to be able to remotely restart any of the gateways.

2.2 Back-End

The back-end consists primarily of a set of data collection tools. These tools may be developed separately and in an ad hoc manner. Back-end tools may be integrated into or deleted from the NMC relatively easily. Additions or deletions can be handled largely by changing the appropriate action routines in the front-end to reflect the change. Of course, the integration of new tools would be aided by prohibiting free form development of new tools. There should be a tool interface standard that all new tools adhere to.

There are two arguments for constructing the back-end out of autonomous tools. This modular structure allows the NMC to gracefully evolve as time passes. The functions of an individual tool need not be pre-defined. The second argument for this structure is that existing network maintenance and monitoring tools could be incorporated into the NMC.

The set of back-end tools need not be constrained to run on the same host. It may be natural for a tool such as a network monitor to run on a separate host from an event logger. Political forces may dictate the hosts that various tools would run on.

3. Protocol

Two levels of protocol are necessary to support the NMC. The primary protocol layer will be provided by a Fault Isolation Protocol(FIP). The basis for the FIP is the Gateway Monitoring Protocol developed for the ARPA Catenet. This protocol would be layered on top of the Internet protocol. The fault isolation protocol is defined in the following section.

Requests that can not be handled by the FIP would be provided for by embellishing the Internet protocol. A Trace option will be added to the Internet protocol. Normally, Internet options may be ignored. Trace is not optional. Trace options must be processed in order that any tool that uses them will function properly. Immediately, this means that all hosts and gateways at M.I.T. will be required to process Trace options. A Trace would require each network node (host or gateway) that sees this request to notify the specified NMC that it had received this trace packet. This request facilitates the location of faulty links (or gateways).

The Internet headers for any FIP packets would be required to have a special protocol number in the protocol field. The protocol number that should be used is 68. This number is tentative until a permanent assignment is obtained from Jon Postel.

Neither the Fault Isolation Protocol nor the Internet Protocol guarantees that all packets will be transmitted reliably. It is desirable that neither protocol requires packet acknowledgements. The lack of packet acknowledgements allows tools like the NMC to be more transparent. Absence of the NMC(s) would still allow the networks to operate properly. As as result of this, the NMC will have to recover from occasional packet lossage.

3.1 The Fault Isolation Protocol

The Arpanet Gateway Monitoring Protocol was originally designed for the Arpa Catenet. The Catenet is a system of connected networks. The FIP will be used to support the NMC in a similar environment. The environment here is the set of connected networks at M.I.T. This environment differs from the Catenet in several ways. The most significant of these is that there is greater control in effecting changes in the constituent parts of the networks. This level of control simplifies the protocol necessary to support network fault isolation tools such as the NMC.

There will be no negotiation mechanism in the fault isolation protocol. Gateways and hosts in the M.I.T. environment will be required to implement the Fault Isolation Protocol.

There will be two classes of packets, requests and responses. Requests are queries to gateways/hosts for specific information. Responses are sent by gateways/hosts in response to a request by the NMC. Responses are also used to send periodic status information to the NMC.

Requests for periodic information are also unnecessary. Periodic information will be automatically sent by each gateway to their local NMC. The NMC is then free to take note of only the information that it is interested in. Since the request class is restricted to one-time requests, there is no provision for the number of times nor the interval between responses.

Bits within fields are numbered from left to right, with the MSB being the leftmost bit. The format of the header for the FIP is shown in Figure 1. The descriptions of the fields appear below:

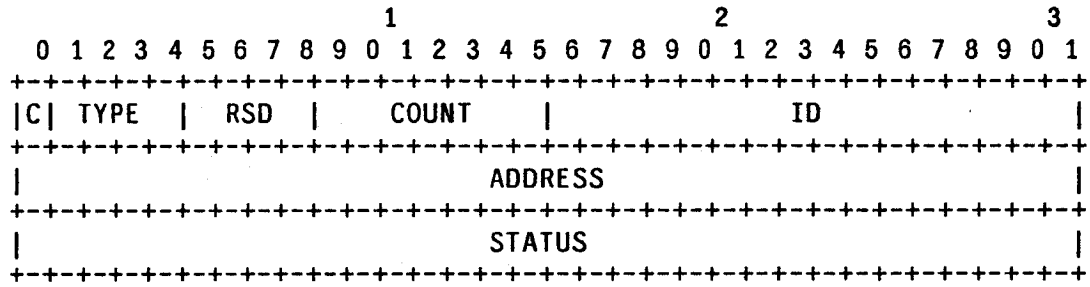
C: 1 Bit
- 0 - Request
- 1 - Response

TYPE: 4 Bits
- 0 - Reserved
- 1 - Gateway/Host Status
- 2 - Trace

Gateway/Host Status indicates that the packet is a request or

---

Fig. 1. FIP Header Format



response for status information of a particular gateway or host. The recipient of a request class packet of type Trace is to send an Internet Trace packet to the destination indicated in the ADDRESS field of the request packet. A response class packet of type Trace is to be used by every host or gateway that sees an Internet Trace packet(including the host that originated the Internet Trace packet).

RSD: 4 Bits

This field is reserved for future use.

COUNT: 8 Bits

The COUNT field is used exclusively in Trace Response packets. It is copied from the COUNT field an Internet Trace packet. COUNT indicates the number of hops an Internet Trace packet had traveled from the host that originated the Trace. It is used to order the Trace Response packets.

ID: 16 Bits

The ID field is used in packet of all classes and all types. It is used to coordinate responses with requests. Responses that have no corresponding request have an ID of zero.

ADDRESS: 32 Bits

The ADDRESS field is used for packets of type Trace to store an Internet address. In a Trace Request packet, the ADDRESS field would hold the destination address that the Internet Trace packet would be sent. The ADDRESS field in a Trace Response packet would be interpreted as holding the Internet address of the gateway or host that sent the Trace Response. Gateway/Host type packets would have zero in the address field.

STATUS: Variable

The STATUS field carries network monitoring information in a Gateway/Host Response packet. All other types and classes of packets would have a zero length STATUS field. The STATUS field options are shown in Figure 2. The format of a STATUS option is shown in Figure 3. The Length field in a

STATUS option is in octets.

3.2 Internet Options

One option, the Trace, will be added to the Internet protocol. The format of this option is very simple. This option is four octets in length. This option consists of three fields. The first field is the type octet. There are three subfields to the type field. The option class will be 2. The option number for the Trace is tentatively assigned to be 6. The second field is the length octet. The third field is the ID. The ID is to be used for coordinating the Trace responses. This is summarized in Figure 4:

Fig. 2. Status Options

Type	Description
0	End of Status Field
1	Gateway/Host Status
2	Connectivity Matrix
3	Routing Tables
4	Queue Length
5	Traffic Statistics

Fig. 3. Status Option Format

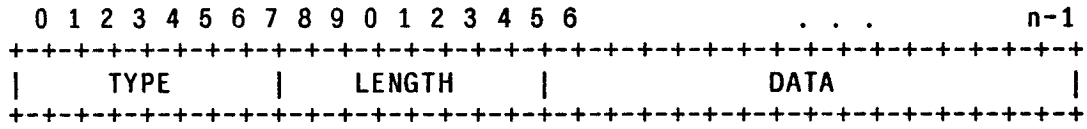
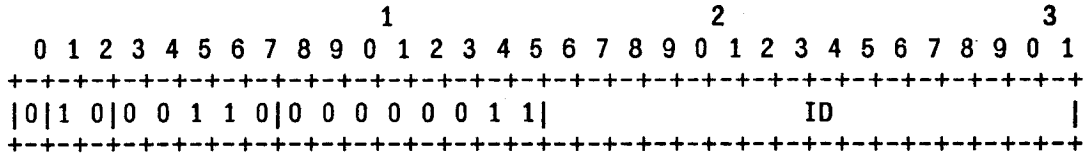


Fig. 4. Trace Option



4. Implementation

A subset of the design presented above will be implemented at LCS using the LCS-NET and the Ethernet. Parts of the NMC that are likely to be built are the front-end and one or more back-end tools.

4.1 Back-End Tools

This section lists several of the more obvious back-end tools that would be useful in the NMC. Some of them are currently under development.

Microprocessor network monitor- Cliff Ludwig is currently working on a Z80 based monitor. It is responsible for keeping an up to date version of what hosts are up on the ring. It would be able to detect is too long and switch out the offending node. It also would be used to switch out LNIs that are faulty, allowing the ring to continue operation. In short, the network monitor is primarily concerned with keeping the ring in operation. The monitor has the power to take out all of the nodes in the network by activating a relay at each node to bypass the node. It would be able to locate faulty nodes by taking out all of the nodes and reconnecting them one by one as long as network continuity is sustained. It would not reconnect any node that would disrupt network continuity.

Auto Restart- This tool is also under development. Auto restart would be provided for gateways and terminal interface units (TIUs). The robustness hardware of the processors of the gateways or TIUs would detect when the processor had failed, and would request that the processor be downloaded from one of the other hosts and restarted. Larry Allen is currently developing this.

Network event logger- There would be an event logger for each subnetwork. The monitor for a ring may seem a likely place for the event logger to reside. For the moment, this is infeasible because the event logger requires secondary storage which is not available on the monitor. It may not be wise to have the event logger reside on the monitor. The Z80 may not be able to collect the required data quickly enough. Also, there may be a valid argument that the monitor is not appropriate because it would complicate the monitor considerably. The monitor has a lot of power over the operation of the ring. Unnecessarily complicating the monitor may cause introduction of more software errors that have the potential of crashing the network.

The types of events that would be logged might include the total number of packets sent over the net during a time period or the number of refused or undeliverable packets sent for each destination host. The event logger would also collect information from the monitor (if there is one) about the LNI's that currently are functioning. The event logger might ask for this information periodically, have the monitor send the information periodically, or have the monitor update the event logger each time the connectivity of the net changes. The event logger would also keep an up to date list of each of the hosts that are operational on the network. It would collect this information by periodically broadcasting a message to each host, asking that each host identify itself. The logger would be notified when the token is restarted. Hosts that are unable to deliver a message to a destination may inform the local event logger that it tried 3 times to deliver a message to some destination and failed. Finally, the event logger would make an entry each time a gateway or TIU had restarted. The gateway or TIU that restarted would inform the event logger as part of the restart process. Each event logged by the event logger would be timestamped as it was collected.

The event logger would not digest any of the information that it collects. The event logger would have to process the information in real-time which would be difficult if not impossible to achieve. By deferring the processing of this information, it allows a degree of robustness to the system. That is, by providing an appropriate program, specific information could be obtained and put into the most useful format. Many such programs could exist for different applications.

Bibliography

[1] Mcdaniel G., *METRIC: a kernel instrumentation system for distributed environments*, Symposium on Operating Systems Principles (Nov. 1977), pp. 93-99.

[2] Page, D.F., *ARPA Catenet Monitoring and Control*, IEN 105, 1979.

[3] Page, D.F., *Gateway Monitoring Protocol*, IEN 131, 1980.

[4] Page, D.F., *The CMCC Terminal Process*, IEN 132, 1980.

[5] Postel, J., *DOD Standard Internet Protocol*, IEN 128, 1980.

[6] Postel, J., *Assigned Numbers*, IEN127, 1980.

[7] Saltzer, J.H., *Environment Considerations for Campus-Wide Networks*, M.I.T. Laboratory for Computer Science memo, 1980.

[8] Saltzer, J.H., *Source Routing for Campus-Wide Internet Transport*, M.I.T. Laboratory for Computer Science memo, 1980.

[9] Vieraitis, R.V., *A Performance Monitor for a Local Area Network*, Bachelor's Thesis, M.I.T., 1980.