UC Irvine
ICS Technical Reports

Title
The ZOTnet : a local area mailing network

Permalink
https://escholarship.org/uc/item/4bm4k2vh

Author
Rose, Marshall

Publication Date
1983-01-03

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

The ZOTnet – a Local Area Mailing Network

Marshall Rose
Department of Information and Computer Science
University of California, Irvine

Mon Jan 3 17:53:53 1983

Computer Mail: mrose%uci@Rand-Relay

Technical Report 200

Introduction

This paper describes the background and implementation of a local area mailing network, the ZOTnet[1], that is currently serving a research environment for the faculty and graduate students of the Department of Information and Computer Science at the University of California, Irvine. Although this paper deals with the framework of the ZOTnet, it discusses several more interesting issues, such as the service model on which computer mailing networks are based, the multi-network environment in which mailing networks must operate, and, the issues of naming and addressing in such an environment. These will be presented in the context of the ZOTnet. Hopefully, this paper will present insight into these issues by examining where the ZOTnet succeeds and fails in its attempt to deal with each issue. Finally, areas where more research is warranted will be presented, along with some ideas that demonstrate how the concepts exemplified by the ZOTnet could prove useful in aiding this research.

Initially, we begin this paper with a discussion of the ZOTnet's architecture. The perspective we will take will be that of the protocols and services that are available in the ZOTnet. In the course of this examination, we shall look at the ISO/OSI reference model and see what similarities can be seen between Open Systems Interconnection and the mailing network.

Next, we will frame the architecture of the ZOTnet in a larger, more abstract view of things, by looking at the so-called UA-MTS-UA model of computer-network mail systems. This model provides the current basis for the explanation of such mail systems. Using it as a backdrop for viewing the ZOTnet should prove to be an interesting experience.

After we have seen how the ZOTnet's architecture contrasts with the OSI model, we shall look at the environment in which the ZOTnet operates. This segment would be very boring if the ZOTnet operated in a closed, static environment. Fortunately, since the ZOTnet interconnects to the NSF CSnet mailing network, which in turn interconnects to the ARPAnet, and from there to the Internet at large, an exciting environment can be expected. In particular, we shall focus on the problems with which interconnection forces us to deal.

Once we have sufficiently examined the complex and dynamic environment that the ZOTnet lives in, we will be prepared to make a penetrating examination of the naming and addressing issues that permeate the ZOTnet and other mailing networks living in the same space. After examining the issues, some solutions will be offered, along with a discussion of how the ZOTnet attempts to implement a partial solution.

Finally, after seeing the ZOTnet from all sides, the reader will be exposed to the ZOTnet's internals. This section will discuss five key components of the ZOTnet that are interrelated to varying degrees. In presenting the components that are running, we shall detail how they fit into the ZOTnet, what services they provide, and how they can be seen in the larger

[1] The name ZOTnet was derived from the U.C. Irvine slogan. The official mascot is the Anteater, which reportedly makes a ZOT sound. Hence, ZOTnet, for UCI's first computer mail network.

UA–MTS–UA context, In presenting the components that are under active examination and thought, we shall identify future research issues.

A (short) note on why the ZOTnet is

Before following the organizational scheme presented above, I feel somewhat compelled to state the underlying philosophy which fueled the ZOTnet effort. Phrased in the simplest of terms, it is believed that interconnection and connectivity are beneficial attributes and are highly desirable in computer systems. If a reasonable degree of connectivity can be achieved cheaply and simply, then the effort to do so would make an interesting research experiment. This is the basis for the ZOTnet. In the latter sections of this paper, the success of the ZOTnet in adhering to this doctrine will be examined in greater detail. Readers interested in pursuing this argument further, should consider [NMA–1] and [NMA–2].

Architecture of the ZOTnet

A well-known view of a layered system is the International Organization for Standardization's reference model of Open Systems Interconnection (ISO/OSI) [IC097-16]. This model views the process of information exchange occurring between autonomous entities as taking place in a seven-level hierarchy. Each level (or layer) in the hierarchy performs a specific set of related tasks as services for the level immediately above it. Each layer accomplishes this by utilizing the services provided by the layer immediately beneath it. Conceptually, the higher the layer is in the hierarchy, the more powerful the set of services that it performs. In specifying OSI, ISO made clear the design strategy that was used in dividing the wide range of services into the seven layers: each layer embodies a particular logical set of activities that are appropriate for the layer's position in the hierarchy.

In addition to the notion that each layer provides services, OSI further views communication as taking place between virtual peers. Each layer has two peers (located at each end of the connection) which are in communication. Notably, the peers are in virtual communication. With the exception of the lowest layer, the peers communicate by using the services provided by the layer immediately below them. Hence, one may view communication as occurring horizontally between peers at the same level of complexity, and vertically between layers in the hierarchy as described in [OSI-1].

Summary of ISO/OSI

In our discussion of the architecture of the ZOTnet, it will be useful to recall the various layers of the ISO/OSI model. The lowest level in OSI is the physical layer. At this point in the hierarchy, physical connections are established, used, and terminated. The communications media directly supplies the means for this. The primary service provided to the next higher layer is the transmission of bit streams between data links. Other services include opening and closing physical connections. Some primitive form of addressing must be present so that the physical layer can establish a connection with the adjacent point with which the higher level wishes to connect.

The level above the physical layer which uses these services is the data link layer. The data link layer is responsible for error-free communication between adjacent links. The data link layer hopes to achieve this by applying sequence, checksum, and acknowledgment information to the real information that it directs the physical layer to move. Error detection is further augmented with a scheme of error recovery, usually involving re-transmissions. In addition, the data link layer is charged with translating the node address to the appropriate address that is understood by the physical layer. Also, the data link layer may inform the next higher level when an adjacent link changes status (e.g., goes down or comes up). Further, the data link layer may also concern itself with limited aspects of flow control, typically using windowing techniques to achieve this.

The layer which makes use of the services provided by the data link layer is the network layer. The network layer, in turn, provides routing services to the next higher layer. That is, when it is told to move information from one node to another, the network layer is responsible for routing that information through any intermediate nodes. In short, the primary service provided by the network layer is end-to-end communication. No explicit guarantee is made that information will be received in the same order as it was originally transmitted, only that some means of setting up a path between two end-points will be used. In order to achieve this, other issues must be considered, such as handling any changes in the network topology, and possibly modifying the routing of information as the load in the subnet changes.

It is now the task of the transport layer to make reliable and economic use of the services provided by the network layer, and to give the next higher layer a simple means of achieving end-to-end communications. Phrased in more precise (and unfortunately) complex terms, this level in the hierarchy provides the next highest layer with a single, logical end-to-end connection, in which information arrives in the same order that it was logically transmitted. Further, the total transmission of information is managed so as to allow a large number of potential users access to the communications network in an efficient fashion. The transport layer makes available several services: an addressing capability which takes a human-meaningful host identifier from the higher layer and maps it into a network-meaningful host number for the network layer; a completely transparent multiplexing of connections requested by multiple instances of the higher layer, and, a mechanism for monitoring and adjusting to changes in the subnet activity (such as congestion).

The level making use of these services is the session layer. The session layer allows co-operating entities to conduct a meaningful session-connection. By this, it is meant that two entities establish a session, conduct business, and then terminate the session. The session layer, in trying to achieve this, is concerned also with high-level synchronization and delimitation. One issue, which is not clearly defined in OSI, is the exact boundary between the session and transport layers. For the purposes of this paper, our interpretation will be that the transport layer concerns itself with system-to-system interconnections, and the session layer concerns itself with process-to-process interconnections.

On top of this, we find the presentation layer which is concerned with the management of structured data. This is the least well defined (and understood) level in the OSI hierarchy. The specification views the presentation layer as allowing the next level to correctly interpret the data exchanged, so perhaps it would be fair to say that the presentation layer makes semantic examination possible by handling syntactic considerations. There is no clear justification for this, however. Other interpretations of the presentation layer, which may support this outlook, tend to view this level as a filter: ASCII to EBCDIC, encrypted to plaintext, and so on.

Finally, we come to the application layer, which is the highest of the seven levels in the OSI hierarchy. The application layer is invoked when a user wishes to use the communication services of the network. At this point in the hierarchy, we have specific applications to achieve, and the application layer is meant to be the top-level entry into the communications network. That is, the application layer is used to give particular applications the ability to communication with other systems.

Framing the ZOTnet

Now that we have been reminded of the particular hierarchy envisioned in the ISO specification of OSI, we can examine how the ZOTnet fits into this view of the world. One primary application is made available to a user in the ZOTnet environment: the capability to originate, examine, distribute, and respond to computer mail. We view these as a single application, although they are in fact very different user activities. In our description of how the ZOTnet compares to the OSI view of things, we shall use the origination, posting, and delivery of a message as the primary example.

To aid our description of the architecture of the ZOTnet, we shall make use of two diagrams. The convention used in drawing these diagrams is for vertical lines to represent the services that a layer offers to the one above it, and for the horizontal lines to represent the protocol that the two peers use to communicate. The term named within a given box is the entity acting as a peer. This convention is followed with the exception that the bottom-most horizontal line is marked with the medium that supports the communication, not the protocol itself.

Before proceeding, recall that the ISO/OSI model is one model of a particular view of the world, hence it should be expected that the architecture of the ZOTnet may not be exactly ISO/OSI.

View One: the ZOTnet domain

Figure 1 pertains to the hierarchy involved when a machine in the ZOTnet communicates with the ZOTnet gateway. At the highest, which would correspond to the application layer in OSI, we find two users conducting a discussion via computer mail. In this particular instance, the node UCI is the ZOTnet gateway, while the node UCI-20A is a machine accessible within only the ZOTnet. The protocol identified is natural language. This is entirely too vague: aside from the technical aspects that the language being used is English text, and not voice or video, it should be noted that a plethora of protocols are included by these two words. Such considerations as etiquette, style of communication, formalisms, and so forth should be represented, but are not. One view might hold that there are actually several levels above the application layer, each with their own purpose, protocol, and service. Since we are concerned with the application of computer mail, we should perhaps take interest in this. It is beyond the scope of this paper to do so, however. Hence, it is reasonable to say that the two users are attempting to hold a conversation, and nothing more.

In order to achieve this communication via computer mail, each user interfaces with a user agent(UA). The service provided by the UA is its user interface, through which the user can direct the origination, distribution, examination, and response of mail. In this instance, the user on UCI is using the Rand Message Handling System (MH), and the user on 20A is using the MM program originally written by Michael McMahon at SRI International. Both are very powerful systems. MM is a single monolithic program, that makes use of the command recognition features of TOPS-20. MH is composed of several relatively simple programs, that share the user's

```
------------------                              ------------------
|  Julian Feldman |      Natural Language        |  David Sheldon  |
|    feldman@UCI |---------------------------- | sheldon@UCI-20A |
------------------                              ------------------

      |                                               |
      | User                                          | User
      |  Interface                                    |  Interface
      |                                               |

-------------------                             ------------------
|    User Agent   |  Internet Message Format    |   User Agent    |
|            (MH) |-----------------------------|            (MM) |
-------------------    Standard (RFC-822)       ------------------

          |                                             |
          | Posting/                                    | Posting/
          |  Receipt                                    |  Receipt
          |                                             |

-----------------    ------------   ------------------
|Delivery Service| Submit | Translate|  Env.  |Delivery Service|
|         (MMDF) |--------| (ZSLAVE) |/////////|       (XMAILR) |
-----------------    ------------   ------------------

     |                       |
     | Channel               |----| Channel
     |  Service                   |  Service
     |                            |

-------------                -------------
|  Polled   |     PhoneNet     |  Passive   |
| Channel   |-----------------| Channel    |
-------------                -------------

      |                           |
      | Physical                  | Physical
      |  Circuit                  |  Circuit
      |                           |

-------------                -------------
| DataSwitch |     RS-232      | DataSwitch |
|            |-----------------|            |
-------------                -------------
```

Figure 1. Interaction in the ZOTnet domain

UNIX[1] environment. Clearly, MH and MM are two entirely different UAs, operating in two entirely different environments, both trying to accomplish the same task. This task is occurring at the OSI equivalent of the presentation layer. Recall that the concern here is for syntactic understanding of information. To achieve this, both UAs have agreed to a common standard for the formatting of messages, i.e., the Standard for the Format of ARPA Internet Text Messages [RFC822].

These UAs do not actually deliver messages, but rely on specific interactions with a message transfer system (MTS) to perform the actual delivery. Two services are required: UAs must be able to post messages with the MTS for delivery, and be able to receive messages from the MTS. The tasks performed by this level are most properly designated as a combination of those occurring at the transport and network layers.

It is important to recall that since the ZOTnet is a mailing network, the application provided is not real-time in nature, but instead is datagram based. In particular, note the instant that a user on UCI wants to post a message for a user on 20A, a connection is not established from UCI to 20A. Instead, the local message service agent (MSA) conducts a well-defined sequence of actions with the UA to ascertain the addressees of the message and the message's text, and then after this sequence has been successfully completed, decides if a connection to 20A should be established. This decision is performed on an addressee by addressee basis, and the MSA may decide to send some addressees their copies immediately, and to wait a while for the others. The decision is made in the context of what channel each recipient's address falls into.

A channel is a logical collection of four items: a site that the MSA must connect to in order to continue delivery, a set of hosts that are served by that site, a protocol that the MSA uses to talk to that site, and a set of channel attributes. For example, UCI theoretically views 20A as being served by the following channel:

<20A, {20A}, PhoneNet, {Active, Polled}>

Which says that 20A is an end-point in the ZOTnet (serves only itself), that UCI can cheaply initiate a connection with at any time (active), and that UCI regularly polls 20A (even though no mail is waiting on UCI). In return, 20A views UCI as being served by:

<UCI, {ZOTnet, CSnet, ARPAnet}, PhoneNet, {Passive}>

Which essentially says that UCI acts as 20A's relay to the rest of the ZOTnet, all of CSnet, and the ARPAnet, and that UCI must be relied on to call 20A, as 20A can not initiate a connection.

The MSA on UCI is the University of Delaware's MMDF system [CSNET-DN3], which forms the basis of the NSF's CSnet PhoneNet project. On 20A, the MSA is XMAILR, a program conceived by Michael McMahon, and developed by Mark Crispin at Stanford University. Neither MSA particularly cares for the other, and the two do not communicate at all. In order to bridge

[1] UNIX is a trademark of Bell Laboratories.

this gap, and achieve interconnection in the ZOTnet, the ZSLAVE[2] program is used. ZSLAVE is a protocol translator that understands the particular (different) protocol used by both MSAs. That is, ZSLAVE takes messages processed by XMAILR for delivery to UCI, and translates them into a protocol for MMDF. Further, ZSLAVE can take messages given to it in the submit protocol used by MMDF, and translate them into the envelope protocol used by XMAILR. The process that actually occurs is rather simple: XMAILR is told that it should communicate with ZSLAVE in order to get all network messages delivered. Whenever XMAILR is given a message to deliver, it delivers to the local recipients immediately, and puts a copy of the message for network recipients in a ZSLAVE spooling area. Whenever UCI initiates a connection with 20A, it starts ZSLAVE and begins using the submit protocol. During the course of this, ZSLAVE is asked if there are any messages for pickup. ZSLAVE scans the spooling area, and translates the XMAILR protocol into submit for each message as it communicates with the invocation of MMDF running on UCI. Similarly, when UCI gives messages to ZSLAVE, ZSLAVE translates the MMDF submit protocol into the XMAILR protocol and places the transformed messages in an area for pickup by XMAILR, and notifies XMAILR that messages are waiting. XMAILR then performs the actual local delivery.

It is now easier to state the actual channel that 20A views as being served by UCI. There are two views — XMAILR's view, and ZSLAVE's view:

<UCI, {ZOTnet, CSnet, ARPAnet}, Envelope, {Active}>
(according to XMAILR)

<UCI, {ZOTnet, CSnet, ARPAnet}, PhoneNet, {Passive}>
(according to ZSLAVE)

At the next level in the ZOTnet, a data link must be maintained between two (topologically adjacent) hosts. The channel in use on each host determines the type of protocol to be used, but the service provided by this level is unaffected: point–to–point channel service. In this example, the PhoneNet protocol [CSNET–DN4] is used. PhoneNet is a simple protocol, which has many deficiencies. It does however get data across a terminal line, check that data for errors, and attempt to recover. The most interesting aspect about PhoneNet is the assumptions it makes about links. In order to run PhoneNet, one needs only to assume a full– or half–duplex terminal line. An operating system views PhoneNet as a user typing lines of ASCII text on a terminal: each packet in PhoneNet is a line of text followed by a CR–LF.

The data link is supported by an RS–232 connection made possible by a Develcon DataSwitch that accesses both the UCI and 20A machines. This is the physical layer in the ZOTnet. The UCI machine uses a special line to the DataSwitch to ask for a connection to the 20A machine, and the DataSwitch selects a terminal port on the 20A machine to make the connection.

[1] ZSLAVE stands for the ZOTnet slave. Since neither of the 2020s in the ZOTnet can initiate a connection, the gateway polls them frequently. The software on the gateway that does this issues commands directing the actions of the program on the 20s: pickup mail, submit mail, or end session.

View Two: the CSnet/ARPAnet domains

Figure 2 pertains to the hierarchy involved when the ZOTnet gateway communicates with a machine outside the ZOTnet. The application and presentation layers are uninteresting in this figure as they have been previously discussed. Hence, we shall begin with the equivalent of the transport layer.

In this instance, we have a user on the ZOTnet gateway holding a computer mail conversation with a user on an ARPAnet machine, Office-8. Since the ZOTnet gateway is not on the ARPAnet, it relies on a CSnet gateway to act as a relay. The current CSnet gateway for the western United States is Rand-Relay, an ARPAnet site. This is especially convenient since the conversation is occurring between a CSnet site and an ARPAnet site, and the Rand-Relay is both. This makes Rand-Relay a internet gateway in the catenet sense [IEN048].

The MSA on both UCI and Rand-Relay is MMDF. The Rand-Relay view of the channel serving Office-8 (or any ARPAnet site[1]), might be:

<OF8, {OF8}, SMTP, {Active}>

The Rand-Relay view of the channel serving UCI is probably:

<UCI, {ZOTnet}, PhoneNet, {Polled}>

As one would expect, the UCI view of the channel served by Rand-Relay is:

<RAND-RELAY, {CSnet, ARPAnet}, PhoneNet, {Passive}>

Hence, mail on the ZOTnet gateway is queued for pickup by Rand-Relay. When Rand-Relay polls UCI, they communicate using the submit protocol over the PhoneNet protocol. This is essentially the same as the communication which occurs between MMDF on UCI and ZSLAVE on 20A. It is after Rand-Relay takes custody of messages from UCI that the interaction becomes interesting.

Rand-Relay establishes an ARPAnet connection over which it conducts an SMTP [RFC821] session with the MSA on Office-8. Rather than being provided with a channel to an adjacent host, a virtual circuit is provided by the ARPAnet transport layer, which uses the Transmission Control Protocol (TCP) [RFC793]. At each end of the connection, sockets communicate, sending TCP segments to each other. These segments are processed as datagrams by the underlying layer, which uses the Internet Protocol (IP) [RFC791] to provide a simple

[1] This points out a deficiency in the way channels are viewed in this paper. The Rand-Relay has a single ARPAnet channel, which is used by connecting to ANY ARPAnet site. There is not an ARPAnet channel for each ARPAnet site.

```
-------------------                              -------------------
|  Marshall Rose  |      Natural Language        | Einar Stefferud |
|       mrose@UCI |------------------------------|        stef@of8 |
-------------------                              -------------------
      |                                               |
      | User                                          | User
      |  Interface                                    |  Interface
      |                                               |
--------------------                             -------------------
|    User Agent    |  Internet Message Format    |    User Agent   |
|             (MH) |-----------------------------|        (HERMES) |
--------------------     Standard (RFC-822)      -------------------
          |                                              |
          | Posting/                                     | Posting/
          |  Receipt                                     |  Receipt
          |                Rand-Relay                    |
------------------         ------------        ------------------
|Delivery Service| Submit  |  Relay   |  SMTP  |Delivery Service|
|          (MMDF)|---------|   (MMDF) |--------|       (XMAILR) |
------------------         ------------        ------------------
     |                        |     |                          |
     | Channel             |--|     |----| Virtual             |
     |  Service            |             |  Circuit            |
     |                     |             |                     |
     |                     |         ----------            ----------
     |                     |         | Socket |    TCP     | Socket |
-----------           -----------    |        |------------|        |
| Passive | PhoneNet  | Polled  |    ----------            ----------
| Channel |-----------| Channel |      | Datagram                |
-----------           -----------      |  Service                |
     |                     |         -----                    -----
     |                     |         |FE |     IP             |FE |
     | Physical            |         |   |--------------------|   |
     |  Circuit            |         -----                    -----
     |                     |           | Packet Service          |
     |                     |         -----    -----           -----
-----------           -----------    |IMP|    |IMP|  IMP-IMP  |IMP|
|  Modem  |   PSTN    | Modem   |    |   |----|   |--- ... ---|   |
|         |-----------|         |    -----    -----           -----
-----------           -----------      |        |               |
                                    -------  -------         -------
                                    |Modem|--|Modem|-- ... --|Modem|
                                    -------  -------         -------
                                           50kbps leased lines
```

Figure 2. Interaction of the CSnet and ARPAnet domains

datagram service. The IP datagrams are in turn processed by the ARPAnet subnet, a large network of IMPs communicating over 50 kbps leased lines.

CBMS view of the ZOTnet

Now that we have seen the actual architecture that implements the ZOTnet, let us consider the logical model to which the ZOTnet attempts to adhere. The ZOTnet is a computer mail network, or phrased more precisely, it is a Computer Based Message System (CBMS). In examining how the ZOTnet can be conceptualized in this respect, we shall use the IFIP Working Group 6.5 model. In examining the relations between the ZOTnet and the CBMS model, it must be emphasised that we shall not consider the functionality enjoyed by the users of a CBMS; rather, we are interested particularly in the model of operation.

The CBMS model presented by IFIP views a CBMS as providing the means for one entity to send a message to another, with very specific restrictions. Each entity interacts with a user agent (UA), and does not engage in direct communication. Similarly, the UAs do not directly communicate, but instead rely on a (common) message transfer system (MTS) to perform the actual transport[1]. Further, the flow of communication is one–way and asynchronous; that is, communication is based on a datagram viewpoint.

A key concept in the IFIP model is that of responsibility. When an originator gives a UA a message for another entity, the UA enters into a *posting protocol* with an MSA. After an agreement is reached that the message has a reasonable chance of being delivered to the recipient(s), the MTS accepts responsibility for the message, and the entity (and UA) relinquish responsibility for the message. At this point, the message is said to have passed through the posting slot, and is now solely in the domain of the MTS. The MTS must now do its best to deliver the message to the recipient's UA. Given that the transit to the locale of the recipient's UA is successful, the message goes through the delivery slot, when the MSA and the recipient's UA enter into a *delivery protocol.* If this is successful, then the MTS relinquishes custody of the message, and it is now in the hands of the recipient's UA.

The model also sees the contents of a message as being two parts: the envelope and the body[2]. The envelope is created by the MSA for the originator, and manipulated by the MTS during the message's transit. In short, the envelope contains whatever information the MTS should need in order to transfer the message. When the MSA at the end–destination gives the message to the recipient's UA, it gives only the message body, not the envelope. The body itself is not examined by the MTS, although information in the body duplicates a great deal of the information kept in the envelope.

[1] The terminology in this paper differs slightly from that in other papers (e.g., [NBS81–5], [X–MHS1], and so forth). Although the term MTS is used for the collection of entities providing message transport services, a single entity in the MTS shall be designated in this paper as a message service agent (MSA), rather than a message transfer agent (MTA).

[2] Between UAs, according to RFC–822, the body is further divided into the headers and the text. Several analogies exist between an MSA's use of the envelope and an UA's use of the headers.

CBMS versus ISO/OSI

It is interesting to note the position taken in [X-MHS1] with regard to the way CBMS protocols fit into the ISO/OSI model. The stance taken is that both MTA and UA protocols reside above the OSI presentation layer (level 6), and are termed message transfer layer and user agent layer (or message processing layer), respectively. As shown in Figure 3, the message processing layer consists solely of the protocols used by the user agents. The UAs themselves supposedly have their channel defined as the sessions held at the message transfer layer between MSAs. This message transfer layer is embodied by the protocols used by the MSAs in the message transport system (see [X-MHS2] for a specification of one of these protocols at the message transfer layer). Finally, observe that the users of the CBMS are *not* viewed in the hierarchy at all!

This is contrasted with the view presented in the previous section as to how the architecture of the ZOTnet could be framed in the context of ISO/OSI. In completing this section, let us attempt a defense of the ZOTnet framing. The ZOTnet architecture tends to place the UAs at the presentation layer. This is consistent with the OSI view that this layer be worried about the syntactic form of information (in our context, the headers of a message), and not the semantic content. The MSAs themselves are placed roughly at the session layer (at the highest point) according to the description of the ZOTnet architecture. Again, since we see MSAs as establishing connections with relays and, eventually, end-destinations, this is easily within the scope of the session layer. Similarly, since some MSAs have the added burden of performing the tasks of the transport layer in choosing the relays to use for a message's transit, we can see MSAs as also being at the transport level as well.

A critical difference in perspective is responsible for the differences between the ZOTnet and [X-MHS1] views. The explanation of the ZOTnet architecture sees users of a CBMS as using that system for a number of applications. On the other hand, the [X-MHS1] view sees all the applications as being provided by the UA. This subtlety explains the difference rather elegantly: in the former, the user is seen as performing applications with the UA, while in the latter, the UA is seen as providing the applications for the user. It is now easy to see how the user and not the UA naturally fits into the domain of the application layer. Along a similar line, since the UAs manipulate the headers of messages, it is simple to understand how they fit in at the presentation layer. This is perfectly consistent with the layered model approach. The message text is used by the higher level (application/user), while the message header is used by the level below that (presentation/UA). It is the philosophy of the ZOTnet architecture however, that the datagram-like nature of the CBMS model allows us the freedom to designate the MTS as performing many of the tasks found in the session and transport layers. This is completely reasonable in view of the fact that the MTS handles the entire transit of a message, a task, which is not unlike those performed by the transport layer.

Finally, it should be realized that the view taken by the ZOTnet architecture is itself flawed. The delays that can occur at the pseudo transport layer can be long (on the order of days), which would not be allowed at the ISO/OSI transport layer. It is maintained though, that the argument is still valid on the basis that everything in the CBMS model is based on datagram-like services, and as such, timing considerations are unimportant (or at least not vitally critical). A CBMS need not be real-time at all. Given this final viewpoint, much support can be given to the perspective of the ZOTnet architecture.

```
--------------------                            -------------------
|    User Agent     | UAs cooperate to provide |    User Agent     |
|                   |--------------------------|                   |
--------------------        CBMS services       -------------------
          |                                             |
          |                                             |
          |                                             |
--------------------                            -------------------
|  Message Service  |      Message Transfer    | Message Service   |
|            Agent  |--------------------------|            Agent  |
--------------------                            -------------------
          |                                             |
          |                                             |
          |                                             |
--------------------/                          /-------------------
|                  Remainder of ISO/OSI hierarchy                   |
|                (starting with presentation layer)                 |
--------------------/                          /-------------------
```

Figure 3. Traditional View of CBMS with respect to ISO/OSI

The ZOTnet environment

Given the architecture of the ZOTnet, and the model that it attempts to fulfill, let us examine the environment that lives in the ZOTnet. Three interconnected domains are considered in this section: U.C. Irvine, CSnet, and the Internet.

U.C. Irvine

The ZOTnet currently serves three research computers operated for the U.C. Irvine Department of Information and Computer Science. Two are DECsystem-2020s running release 4 of the TOPS-20 operating system. In the ZOTnet (and CSnet) domains, these are officially designated as UCI-20A and UCI-20B. These are passive sites in the ZOTnet. As discussed in the section on the architecture of the ZOTnet, neither 20 has the capability to establish a connection. The third machine is a VAX-11/750[1] running 4.1bsd UNIX, with the official designation as UCI-VAX[2]. The VAX acts as the gateway for the ZOTnet, and thus has the name UCI in addition to its name of UCI-VAX. The current polling strategy for the gateway is to view each 20 as active and polled. By this, it is meant that a given 20 gets polled about every 10 minutes and, in addition, will make a connection whenever the gateway gets mail for a given 20. Further, the two 20s compete for the gateway's service. A single background process on the VAX is devoted to keeping the 20s polled, further only a single terminal line is dedicated for the ZOTnet connection.

UNIX and TOPS-20 provide two radically different user environments to the researchers in the ZOTnet. Each research group tends to base itself on a single machine, which is best suited for their activities. If members of a given research group were to communicate only with other members of their particular group, then a case might be made that the utility of the ZOTnet was in question. Nothing could be further from the truth. Members of different groups do communicate, quite heavily in fact[3]. In addition, this argument does not even consider the fact that researchers in the ZOTnet also maintain lines of communications with researchers at other sites.

[1] VAX is a trademark of Digital Equipment Corporation.

[2] This is for purely historical reasons. There are, at the time of this writing, two VAX-11/750s and one VAX-11/780 on campus Since the UCI-VAX has the first VAX on campus however, it exercises squatter's rights in the naming domain.

[3] The sanctity of mail in the ZOTnet is taken very seriously and, as a result, recording the usage of computer mail is a extremely sensitive issue. The most granular amount of information that has ever been gathered in the ZOTnet was to keep track of the number of messages and their sizes on a machine by machine basis. No information was ever kept on who was sending mail, or to whom.

Prior to the installation of the ZOTnet, if two correspondents were on different machines, they either did not communicate by computer mail, or one had to gain access to a guest account in order to send mail to the other. To meet this problem, in a naive sort of way, the department authorized a proliferation of logins on each machine, so that corresponding parties could use computer mail. The result of this, in addition to all of the tedious administrative overhead, was that researchers ended up with more than one mailbox. Computer mail was constantly getting lost, not by any fault of the transport system, but simply because it had the misfortune of arriving at the wrong mailbox! This tended to give researchers the view that computer mail was too much of a bother, since one has to keep checking all these mailboxes, on different systems with different environments. In response, either a lot of time was wasted checking a lot of mailboxes, or mail did not get read. Regardless of the individual choice made, it was a major loss.

Now that the ZOTnet gives the researchers in the department a good degree of interoperability on the research machines, the situation has improved dramatically. A very important service that the ZOTnet provides is its naming domain, and the concept of a mail–handle. These will be discussed in the next section.

One weakness of the ZOTnet, is that full interconnection has not been achieved. In addition to the concern that the 20s are passive sites (which may or may not be a problem, this issue is mostly political), the fact still remains that the ZOTnet supports neither a file transfer facility, nor a virtual terminal facility. Networks that have achieved full interconnection, such as the ARPAnet, do provide these services, and they are used extensively. A CBMS cannot provide these due to its datagram–like nature. Despite this inherent limitation, the ZOTnet does achieve a primitive file transfer capability. Users in the ZOTnet often send papers and other plain–text objects to correspondents by wrapping the object up in a message. Each machine in the ZOTnet has at least one UA that supports an Insert File command, so it is relatively easy for users to do this. In short, file transfer in the ZOTnet is built on top of mail[1].

CSnet

Now let us consider the additional complexities that are involved when the CSnet domain is included. Access to CSnet provides the users of the ZOTnet with access to two domains: the general CSnet domain and the larger ARPAnet domain.

CSnet is a logical binding consisting of certain hosts in the ARPAnet, Telenet, or PhoneNet that agree to a common set of hardware, software, and protocol configurations [CSNET–DN1]. CSnet came about as an effort to meet problems in the area of Computer Science. CSnet hopes to solve some of these problems by providing by allowing researchers to communicate easily, and to access information [CSNET1].

The backbone of CSnet is a powerful MSA known as the Multi–Channel Memo Distribution Facility (MMDF). As of the time of this writing, the CSnet domain supports computer mail as its only application. CSnet is rapidly growing in both size and functionality, and hopes to provide other services to its member sites, such as remote database and resource access, a CSnet nameserver [CSNET–DN2], and file transfer capability.

[1] This is contrasted with the way that mail was first implemented in the ARPAnet. Until introduction of the Internet SMTP (Simple Mail Transfer Protocol) in the ARPAnet, computer mail was built on top of the ARPAnet FTP (File Transfer Protocol).

One very powerful aspect of MMDF, is its conceptualization of channels, which views channels as protocol fitters, and provides for the capability to achieve interconnection between different networks. This has proved most useful with respect to forming gateways between the CSnet and ARPAnet domains. Such a gateway consists of a CSnet host that is in the ARPAnet, acting as a store–and–forward relay. Messages enter the gateway's queues through one channel, and leave through another. An example of this was shown in Figure 2.

ARPAnet

Finally, let us consider how the ARPAnet domain affects the ZOTnet environment. As discussed above, CBMS access to the ARPAnet is achieved in the ZOTnet by virtue of interconnection to CSnet and the existence of a CSnet/ARPAnet gateway. Again, the same types of CBMS service provided by the CSnet connection are found with the ARPAnet connection. Only one additional complexity is added, namely, that of the host space. ARPAnet sites do not generally recognize CSnet[1] sites; that is, the names of CSnet sites are not normally found in ARPAnet host tables. Although it is beyond the scope of this paper to discuss the politics of this policy, the fact remains that it would be a real problem to keep all the ARPAnet sites advised of the names of all the CSnet sites.

The result of this would be that while mail can flow from the ZOTnet to the ARPAnet, it cannot flow from the ARPAnet to the ZOTnet, and the mail flowing from the ZOTnet to the ARPAnet is not replyable. Fortunately, this is not the case. Let us examine why this problem could come about, and how it is avoided. The problem centers around two issues: address verification, and message routing. Although address verification schemes vary, we shall consider a rather common approach which, because it is highly practical, enjoys wide use. This method asserts that the validity of an address depends not only on the address itself, but on the site that is trying to verify the address. That is, a fully qualified address may be valid at a given site, but invalid at another.

Consider the following definitions:

Host Space: in the context of a given host, the combination of all domain spaces known to that host.

Domain Space: in the context of a given CBMS network, the names of all the sites in that network.

Address: in the context of a message, an ordered pair <mailbox, host>, describing the end–destination where a copy of that message should be delivered.

Simply put, to verify an address, compare the host portion of the address to the name of the site that is performing the verification. If they are the same, the address is termed local; if not, then the address is termed non–local.

For local addresses, use some method to see that the mailbox portion of the address is valid. Usually, the mailbox is the login name of a user, and this verification is simple. In other cases, the name of the mailbox may refer to a mail alias. In cases such as these, the address is valid, and the MTS propagates copies of the message for each address in the alias. The semantics

[1] Obviously, CSnet/ARPAnet gateways are an exception.

of this are discussed in the next section.

For non-local addresses, a check is made to see if the host portion is found in the site's host space. If found, then no further verification of the address is required. This is a direct consequence of the datagram-like nature of a CBMS. It would be extremely costly to verify the address further, because it would require that each site keep a list of all valid mailboxes for all sites, or that each site be able to query all other sites as to the address' validity. Neither is currently practical.

It is now clear, given that the ARPAnet sites do not include ZOTnet (or CSnet) sites in their host space, all ZOTnet addresses (which would be completely valid in the ZOTnet domain) are guaranteed to be invalid in the ARPAnet domain. If it is taken for granted that being able to send a message to someone and having the recipient being unable to reply is down-right silly, then this could be a tremendous problem.

To consider how this problem is avoided, it is required that a clear distinction be made between names, addresses, and routes. For simplicity's sake, let us denote a name as the title of an object, an address as the location of an object, and a route as the path by which a message might travel to reach the addressed location. Typically, a CBMS user is concerned with only names and addresses, particularly the latter. It is possible though, to conceal routing information in the address. This technique is known as source routing. Source routing is supported by MMDF, the CSnet MSA, and is also supported by the ZOTnet MSAs. Source-routed addresses tend to take the form:

mailbox%host1%host2%...%hostN@site

which says the route that the message should take in order to get to this address is to first go to site, from there to hostN, from there to hostN-1, and so forth, until it finally arrives at host1.

It is now easy to see how a user in the ARPAnet domain addresses messages for users in the ZOTnet (or CSnet) domain:

mailbox%host@gateway

where gateway is the name of a site that acts as a CSnet/ARPAnet gateway, and mailbox@host would be a valid ZOTnet (or CSnet) address. This solves the problem of ARPAnet users originating mail for the ZOTnet, providing they are careful to use this form of addressing. But what of the addresses in the messages that originate from the ZOTnet? In particular, when a user in the ARPAnet domain receives a message from one in the ZOTnet domain, one would suspect that the addresses in the message would not be valid. The activity that will demonstrate this easily is the reply command. All reasonable UAs should permit a user to reply easily to a message sent by another, by creating a new message, the reply, whose headers are constructed by examining the original message. Generally, the addresses for the reply are taken from any existing "From:", "Reply-To:", "To:" and "CC:" fields of the message's header. If any of these addresses are in the ZOTnet domain, they will be invalid in the ARPAnet domain. In order for the reply to be successful, the user will probably have to edit these automatically constructed headers, or originate a new message.

The solution to this is to have the CSnet/ARPAnet gateway do header munging[1]. Essentially, a gateway address munger violates the CBMS rule for MSAs and looks at the headers of the message (recall that the headers are in the body of a message and not the envelope). For each header which should contain addresses, the MSA examines each address. If the host portion of an address is not in the domain of the CBMS the gateway is giving the message to, then the gateway will munge the address. The munging that occurs is simple: mailbox@host becomes mailbox%host@gateway. It should be noted that this discussion is not particular to the ZOTnet. Any network connected to CSnet but not known to the ARPAnet could be discussed in these terms.

[1] This term is somewhat less technical and more accurate than many would care to admit.

Names and Addresses

We have now briefly considered the environment of the ZOTnet, and the interactions which occur with the CSnet and ARPAnet domains. It is time that we consider how names and addresses interact in the ZOTnet. A very painful experience that is encountered when interconnection is achieved in a CBMS environment is the address problem: if you know that someone resides in a given domain, how do you address mail to go to that person? Phrased another way, how does one easily map names into addresses?

For the ARPAnet domain, this problem is solved by a Name Server. If you know someone's name, and they have registered themselves in the NIC's (Network Information Center) database, you can run a program that establishes a connection with NIC and queries the database. In a strictly CBMS environment, this is not possible. Further, users in the ZOTnet are not registered with the NIC. Hence, users in the ARPAnet domain will not be able to successfully use the NIC name server to get the address of a user in the ZOTnet domain.

The Mail-Handle Concept

It turns out that the ZOTnet's (relatively) small size[1] is an advantage to solving the problem. All users in the ZOTnet domain, upon receiving or terminating a login, conduct an administrative transaction with the ZOTnet PostMaster (whether they know it or not). Each user in the ZOTnet is assigned a *mail-handle,* which can be viewed as the CBMS equivalent of the user's real-world name. Since all users are so registered with a single naming authority, mail-handles are unique within the ZOTnet. Although the policy for generating a mail-handle from a user's name is not set in stone, the following rules are consulted when mail-handles are generated in the ZOTnet:

1. If possible, a person's mail-handle is the person's last name. This may not be possible if there is more than one user in the ZOTnet with the same last name, or if this is a likely event.

2. Failing that, if possible, a person's mail-handle should be the concatenation of the first letter of the user's first name with the last name (e.g., MRose).

3. Finally, if the introduction of a mail-handle would duplicate another person's mail-handle, then a different mail-handle must be obtained, and the mail-handle for the

[1] Even if the ZOTnet were a trivial network (i.e., one host), the problem would still exist. One might know the person's real-world name (e.g., John Mangrich), but not his mailbox address (e.g., grich%uci-vax@Rand-Relay). In this particular instance, the local host's restrictions on the length of a login identifier prevent the user's entire name from being used. This is just one of many examples why the name of a person may not be the person's local address.

original owner must be changed.

The reason for the third rule is somewhat counter-intuitive. After all, if a person is known by a mail-handle for quite a while and then this mail-handle is changed, those corresponding with the user will be in for a rude surprise. After getting some failed mail notifications from the ZOTnet, they will realize that the address they were using is no longer valid, and that they will have to find the correct mail-handle if communication is to continue. This could result in having to go out-of-band to find the correct address. In most situations however, the user is given adequate notice, and frequent correspondents are apprised in advance. The reason the third rule is followed is because it is felt that one thing worse than getting messages back is to receive mail meant for someone else. Again, the sanctity of the mail is a prime consideration in this value judgment.

If it is given that mail-handles are reasonably chosen, then they provide an alternate naming space for users in the ZOTnet. That is, a CBMS user who can reach the ZOTnet need not worry about what machine a recipient is based on in the ZOTnet; rather, the CBMS user need only know the recipient's mail-handle. The aliasing system in the ZOTnet will take it from there. In one sense, it can be said that the ZOTnet allows names to be used instead of addresses! Further, it should be noted that users in the ZOTnet domain enjoy these advantages, they need not worry what machines and mailboxes of other researchers, rather, they need only know the mail-handle.

Mail-Handles versus Addresses

It has been shown how names map into mail-handles, so now let us consider how mail-handles map into addresses. Since each login in the ZOTnet is registered, all primary mailboxes for a given person in the ZOTnet are known. The ZOTdb system keeps track of all users, mail-handles, and mailboxes, and generates forwarding tables for each machine in the ZOTnet. When an address is verified, these tables are consulted, and the appropriate mailboxes substituted. Naturally, the most important forwarding table resides on the gateway, as it is the one consulted when messages from other domains are encountered.

With this in mind, let us consider how a user on the ARPAnet Office-8 machine (OF8) sends mail to a user in the ZOTnet. For this example, let us say that the user actually resides on UCI-20A, but the ARPAnet user does not know this. The user in the ARPAnet domain addresses the message to mail-handle%uci@Rand-Relay. The OF8 host knows that it is not Rand-Relay; the address is non-local. Further, Rand-Relay is in the ARPAnet domain, so the address, as far as OF8 is concerned, is valid. OF8 gives the message to Rand-Relay. The Rand-Relay host knows that the address is local, so the Rand-Relay MSA tries to validate the mailbox part, mail-handle%uci. Seeing the source-routing, it sees if it knows about UCI. UCI is in the CSnet domain, which is in Rand-Relay's host space, so it accepts the address as valid, denoting it as mail-handle@uci. Rand-Relay gives the message to UCI. The UCI host knows that the address is local, so it tries to validate the mailbox part, mail-handle. Upon consulting the ZOTnet forwarding table, it sees which mailbox the mail-handle maps to, so it accepts the address as valid, denoting it as login-name@UCI-20A. UCI gives the message to UCI-20A. The UCI-20A host knows that the address is local, so it tries to validate the mailbox part, login-name. Following this, the message is delivered.

Realize however that aliasing is a very seductive and dangerous trap. In the domain of the ZOTnet, where the aliasing is under strict control, few problems (if any) are encountered. But, in other cases where aliases are used, many problems can be encountered. All of these stem from the fact that as addresses change, aliases become outdated. Further, one should note that the MTS is manipulating the message envelope, not the message body. The mapping from mail-handle%uci@Rand-Relay to mail-handle@uci to login-name@uci-20a all occurred inside the message's envelope. When the user on 20a gets the message, the headers show the original address that was used in the ARPAnet domain, and none of the translated addresses.

Aliasing can be used for purposes other than mail-handles, they can be used for mass-distribution lists instead. The difference between the two is the type of mapping that occurs: with mail-handles the mapping is one-to-one, with distribution lists the mapping is one-to-many. That is, suppose someone sends a message to a distribution list, which on the surface looks just like an address, say PROTOCOLS@RUTGERS, as an example. When the message gets to the RUTGERS host, the local mailer notices that PROTOCOLS is an alias, and expands it. In this case, let us say that the PROTOCOLS distribution list maps to a few hundred other addresses, some on other hosts, any of which may further be aliased. As a result, the local mailer then propagates copies of the message to each address in the alias. Problems are encountered when one of these addresses or an address from a future alias replacement is bad. If a recipient has moved or has gone away, the message cannot be delivered. The result is that the *originator* of the message, who naively sent it to PROTOCOLS@RUTGERS, and not the *maintainer* of the PROTOCOLS distribution list, gets the failed mail notice.

The reason for this lies in the way that most MSAs use the envelope. Conceptually, we may view an envelope as consisting of two parts: the first is the sender of the message, the second is a list of recipients[1]. As the message is aliased and delivered, the second part changes, and when it finally becomes null, the message is deemed delivered. The first part is never changed however, even when aliasing does occur. This introduces the obvious question: Why doesn't the sender change when addresses in the envelope get changed? There are two reasons: first, it is unclear as to who the sender should be changed to, and, second, there are some instances where it might be desirable for the originator to be notified.

In order to answer these concerns, let us rely on the distinction made between types of aliasing. In the case of mail-handles, clearly, we want the originator to know that the mail-handle is no longer valid, and the failed mail notice that is received should contain a summary of the problem and a copy of the message that failed, as well. Also, the maintainer of the mail-handle (e.g., the PostMaster), should be notified, but that notice should only include the problem summary, along with the address of the originator. A notice is required, since the failure of a mail-handle may indicate a grave problem in the MTS. But, because we are concerned with the sanctity of the mail, if the PostMaster requires more information than the address of the originator, the mail-handle, and a problem summary, then the only proper recourse is to ask the originator for more information. In the case of distribution lists, the originator probably does not care if some propagations of his message were not delivered, but the person who maintains the distribution list does. Hence, a problem summary containing the address of the originator, the location of the distribution list that produced the bad address, and the bad address should be sent to the maintainer of the distribution list.

[1] There is usually also a time-stamp indicating the date and time that the message entered the MSA's queue.

The point of all this is that there is responsibility involved when aliasing is used, and that the MTS must address itself to this problem. Unfortunately, keeping track of all this information in the envelope is an entirely new problem. Since each address can presumably be aliased once it reaches its local site, we really require two attributes for each address in the envelope: a flag, telling whether the originator should get a full failed notice back, and the address' sentinel, which is an address telling who should get the failure summary for the address if it turns out that the address is undeliverable. The flag would be set to TRUE if the address was the result of a one-to-one mapping (mail-handle) or FALSE otherwise (distribution list). But, suppose the address is bad and we try to return a failure summary to the sentinel, what if the sentinel's address is bad? Finally, realize that all this presumes the MSA can deduce the address of the sentinel. Both these problems can be solved, but this scheme can be implemented only if all MSAs in the MTS agree to this new envelope policy. Since this adds an entirely new level of complexity to the addressing problem, it is somewhat unlikely. Fortunately, there are other solutions to the problem.

UA versus MSA

In order to solve this problem, we might find it useful to shift the burden of handling distribution lists. In short, the methods currently used are inadequate, and fixes to these inadequacies cannot be easily made in each MSA and universally implemented across the MTS. Perhaps this should be a task for the UA then and not the MSAs? What is lost if this switch in responsibility is made?

One highly visible difference is that of automation. Note that the MTS tends to be viewed as rather automated: there is typically no user *directly* guiding its actions. While it is true that the activities in which the MTS engages are a direct consequence of the activities of users, a single user is not the focus of attention. With a user agent on the other hand, conceptually, we envision a user interactively running a user agent as being the focal point. This conceptualization is incorrect. Speaking strictly from the perspective of the CBMS model, the dividing lines between user agents and the MTS are the posting slots and nothing else. There is nothing in the model to prevent the user agent from acting without a user directing it.

This is in fact a good metric for seeing how advanced a particular CBMS is. If the CBMS supports user agents who can act independently of the user's presence, then the CBMS has the potential for being much more powerful than a CBMS which does not support such user agents. This is, of course, a very subjective statement, but it will be substantiated by the end of this section. A very common type of user agent that runs independently of the user is known as the receive-mail hook. This hook is invoked by the MSA on the user's behalf whenever a message is about to pass through the delivery slot. The message is made accessible to the hook, and the hook may, in some limited ways, direct the final delivery of the message. Depending on how much information is presented to the hook, it is possible that the hook could serve as a re-distribution entity. The operation would be relatively simple: upon invocation, the hook would determine which distribution list it was serving, and it could then post the message with the transport system, with some modifications. First, it would mark the person responsible for the list as being the sender. Hence, all failed-mail notices would go to the maintainer and not the originator. Second, it could add whatever distribution information that would be appropriate to the message, such as unique identifiers, "Reply-To:" fields, and so forth. Third, it could save a copy of the message for the maintainer's archives. The possibilities are endless.

Note the key advantage achieved here is one of flexibility. Changing each MSA so that the entire MTS could support all of these functions would be impossible, but, depending on the power of each individual CBMS, distribution lists could be handled properly. A case might be made that the MTS should be concerned only about transport and not re–distribution. Providing that distribution lists were prohibited, and mail–handles were permitted, then this would be a worthy goal. Unfortunately, as soon as any aliasing capability is permitted, which would be required for mail–handles, then distribution lists are also possible. Because of this, each PostMaster would have to be relied upon to uphold this policy.

Implementation of the ZOTnet

It is now time that we view how various parts of the ZOTnet are implemented, and their underlying rationale. We shall begin with ZSLAVE, which makes interconnection possible in the ZOTnet. Second, we shall look at how mail–handles are dealt with in the ZOTnet, and the ZOTnet database, ZOTdb, that maintains them. Next, we shall see how large distribution lists are handled within the ZOTnet. Finally, two issues of current research in the ZOTnet will be identified: message archive handling and high–bandwidth message handling. The discussion of these implementations will not be as technical as one would expect. In particular, more attention will be paid to the algorithms used and less to history and background.

ZSLAVE

As noted in the section on the architecture of the ZOTnet, ZSLAVE is the name used for the ZOTnet server on the 2020 systems. Since the 20s are not active sites, ZSLAVE is invoked by the VAX, and does not establish connections itself. ZSLAVE handles two levels of protocols in the ZOTnet; it acts as a protocol translator for the MMDF submit and XMAILR envelope protocols, and it also implements the PhoneNet protocol for data link level activity. For the former, ZSLAVE takes commands from its counterpart on the VAX telling it what type of activity is to take place. There are currently three commands defined: SUBMIT, PICKUP, and END. For the latter, the program on the VAX, ch_phone, and ZSLAVE first negotiate critical PhoneNet parameters and then pass packets back and forth.

Figure 4 shows that the organization of the ZSLAVE consists of two main peers whose actions are coordinated by a series of top–level control sequences[1]. Each peer consists of an MSA layer and, possibly, a data–link layer. Each layer is generally composed of several sub–layers, which can be thought of as hidden subroutines.

The ZPHONE/ZDIAL modules are essentially mirror images of the corresponding routines in the process running on the VAX. ZPHONE concerns itself with the MMDF MSA protocol submit. It takes the data portion of packets given to it by the ZDIAL module, and does the appropriate processing. ZPHONE also gives the ZDIAL module data to send across the connection back to the other MSA. The ZDIAL module fully implements the PhoneNet protocol.

The ZMAIL module is much more interesting. Although it has routines that are very similar to ZPHONE, it is a protocol translator. It takes XMAILR envelopes and translates them into data for the VAX, or takes data from the VAX and translates them into envelopes for XMAILR. Fortunately, the differences between the two MSA protocols are not so great as to prevent translation, although the ZMAIL module does have to allow for several subtleties in both.

[1] There are actually other modules which perform logging and so forth, but they are uninteresting in this context.

```
               ZSLAVE                   <- session protocol
                 |
       ----------------------
       |                    |
     ZMAIL               ZPHONE         <- MSA protocol
                            |
                          ZDIAL         <- data link protocol

     XMAILR               MMDF
    interface           interface
```

Figure 4. The organization of ZSLAVE

ZSLAVE itself is not a research topic, although some of the principles it embodies are. In particular, protocol translation is not well understood and enjoys much less usage than protocol encapsulation. Research into the translation and encapsulation aspects of fitting protocols should prove quite interesting.

Mail–Handles

Each user in the ZOTnet has a mail–handle that acts as a binding between the user's name and address in the ZOTnet. Although simply stated, the implementation of mail–handles is not trivial. Users may have more than one primary mailbox that they may wish their mail–handle to affect. Further, users may also be known by more than one name (true aliases). Finally, mail–handles change occasionally and logins change much more frequently, so the naming space is dynamic. It is indeed fortunate that a central naming authority exists in the ZOTnet, so that ZOTnet's database (ZOTdb) can be given consistent information as to the status of users, logins, and mail–handles. In examining the implementation of mail–handles, we shall consider both how users are viewed by the ZOTdb, and how the database generates the forwarding tables. That is, we will be concerned with both representation and processing.

The ZOTdb knows about two different types of entities: sites and users. Sites are hosts in the ZOTnet, and users are people who use the hosts in the ZOTnet[1]. The structure of sites is rather unspectacular: sites have a name, some flags describing the local MSA, and a gateway. The significance of the gateway will be discussed later. Users, on the other hand, have a much more interesting structure:

Personal Name: the real–world name by which the user is known.

Mail Handle: the binding between a user and the user's address.

Aliases: a list of common synonyms for the user's mail–handle which are generally common misspellings for the mail–handle, or nicknames for the user.

Machines: a list of machines on which the user has logins. One of these is designated as being the home machine for the user in the ZOTnet.

> *Machine Name:* the site name of the host in the ZOTnet.
>
> *Routing Information:* a list of ZOTnet addresses (machine–mailbox pairs) for the user on this machine to re–route mail to.
>
> *Special Information:* a list of other addresses (not in the ZOTnet) that mail for the user on this machine should be re–routed to.
>
> *Status Flags:* information describing to what extent this machine participates in forwarding.
>
> *Mailboxes:* a list of logins the user has on this machine. One of these is designated as

[1] Actually, it is possible to be known by ZOTdb, but not to be situated on a host in the ZOTnet. This mechanism is used for faculty who are on leave and may be monitoring computer mailboxes in other domains.

being the home mailbox for the user.

Mailbox name: the name of the login.

Routing Information: a list of ZOTnet addresses (machine–mailbox pairs) that mail for this mailbox should be re–routed to. This supersedes the machine routing for this particular mailbox.

Special Information: a list of other addresses (not in the ZOTnet) that mail for this mailbox should be re–routed to. This supersedes the machine special for this particular mailbox.

Status Flags: information describing the extent to which this mailbox participates in forwarding. These may possibly supersede corresponding flags for the machine on which this mailbox resides.

A small database program is used by the ZOTnet PostMaster to keep the database updated, and several utility programs exist which automatically generate the forwarding tables for the PostMaster. Let us now consider how the forwarding entries for a given user on a given machine are generated. Figure 5 presents a subset of the generation algorithm.

Basically, the algorithm performs three steps. In the first step, each mailbox the user has on a given machine is resolved. If the mailbox explicitly is not to have mail forwarded, then it is not mapped to the user's mail–handle. Independent of this, if the mailbox is to have special routing done for it, then these mappings are made as well. At the end of this step, each mailbox has been accounted for. Hence, if a user wishes to work in several logins, but wishes to read mail only from a single mailbox, this is a very simple task. If a user wishes to have copies of the mail directed towards certain mailboxes re–distributed automatically, this is handled simply as well. Finally, if a user wishes to have the ZOTnet not direct mail for the mailbox, this is handled as well.

In the second step, each alias for the user's mail–handle is resolved. In doing this, mailboxes are given precedence over aliases. This avoids the ambiguities that can arise if the user's mailbox has the same name as the alias. Since aliases are valid across the ZOTnet domain, the naming space for the local host is given precedence.

In the third step, the user's mail–handle must be resolved. This step is somewhat tricky, and requires some comment. At this point, all of the mailboxes for the user have been mapped to either special routes (which are independent of the mail–handle), or the user's mail–handle. Further, all aliases for the user in the ZOTnet domain have been mapped to the mail–handle. Hence, in order to close the naming space, we need to resolve the mail–handle. If the user has at least one mailbox on this machine, then it is possible for the mail–handle to map to one of these mailboxes. The choice is strictly that of the user. Instead, if the user wishes all mail for the machine to be directed elsewhere, then routing information can supersede the mail–handle. Finally, if there is any additional routing to occur for the user on this machine, this is mapped for the mail–handle as well.

If the user did not have a mailbox on this machine, then a distinction must be made as to whether or not this machine is the ZOTnet gateway. If it is not, then we can postpone the actual binding of the mail–handle by mapping the mail–handle at this site to the mail–handle at the gateway. With the gateway however, we must make the decision as to where the mail–handle

```
                /* resolve mailboxes */
FOREACH mailbox (if any) on this machine that the user has:
   IF this mailbox is participating in forwarding,
      IF this mailbox has routing information,
         map this mailbox to the routing.
      ELSIF
         map this mailbox to the user's mail-handle.
      ENDIF
   ENDIF
   IF there are any special addresses for this mailbox,
      map this mailbox to them as well.
   ENDIF
ENDFOR

                /* resolve aliases */
FOREACH alias (if any) that the user has in the ZOTnet domain:
   IF the alias is not the same as any of the mailboxes
   that the user has on this machine,
      map the alias to the user's mail-handle.
   ENDIF
ENDFOR

                /* resolve mail-handle */
IF the user did have at least one mailbox on this machine,
   IF this machine is participating in forwarding,
      IF this machine has routing information,
         map the user's mail-handle to the routing.
      ELSIF
         map the user's mail-handle to the home mailbox.
      ENDIF
   ENDIF
   IF there are any special addresses for this machine,
      map the user's mail-handle to them as well.
   ENDIF
ELSIF
   IF this machine is the gateway,
      map the user's mail-handle to the home mailbox
      on the user's home machine.
   ELSIF
      map the user's mail-handle to the gateway.
   ENDIF
ENDIF
```

Figure 5. Subset of the Forwarding Generation Algorithm

must map. Recall that the aliasing that occurs at the gateway is also used on all messages entering the ZOTnet domain, hence it is critically important that this table be up–to–date. The mapping that occurs at the gateway is rather simple minded: we map the mail–handle to the home mailbox on the home machine for the user.

There are two problems that can occur when such heavy aliasing is done, even within the limited confines of the ZOTnet. First, it is possible that some forwarding equations may be degenerate (e.g., of the form X maps to X). Since this is observable at the time that the forwarding tables are generated, such equations are removed easily. Second, it is possible that aliasing loops may occur between machines. These are potentially dangerous, as once a loop is established, all mail in the loop is trapped! Once a loop has started, the only possible way to break it is to update a forwarding table contributing to the loop in such a manner as to direct the mail to a mailbox. Detecting such loops when they occur is not easy, as mail is usually re–directed, and not propagated. As a result, a small flow of mail seems to be in the system constantly, and does not noticeably affect the bandwidth. Such loops should be detected when the tables are generated, in addition to implementing mechanisms to detect loops when they are exercised. This could be done, providing that the ZOTdb is located on a single host, and all tables generated can be checked simultaneously.

These two problems remain as research topics. It is interesting to note that the traditional methods used by the network layer in managing packet routing prove to be only partially useful. Both time–stamps and hop counts provide some insight but are not complete. The alias looping problem differs significantly inasmuch as messages can propagate as well as be re–directed, and the mechanisms in the ZOTnet should not prevent their legitimate use of this.

Distribution Lists

Earlier in this paper, a case was made for having distributions being handled by UAs rather than the MTS. In the ZOTnet, both entities are used, to meet different problems. Bandwidth in the ZOTnet is a scarce resource. Since the gateway uses the PhoneNet protocol, it is currently limited to 1200 baud connections when it interoperates with other sites (both inside and outside the ZOTnet). Since distribution lists imply more than one recipient, an active discussion group with a popular following could easily overrun the ZOTnet's channels. Fortunately, message envelopes can have more than one addressee and as such, the bandwidth can be greatly conserved: A single copy of a message for each distribution list enters the ZOTnet.

This does not address the problem as to what happens after the message is in the ZOTnet. In particular, discussion groups are interesting to the users on more than one machine, and each machine has more than one user interested. As such, the gateway distributes a copy of the message to each of the other machines in the ZOTnet. In order to prevent a single message from propagating into a message for each individual user, the ZOTnet designates a special area on each machine as a BBoard holding area. All major distribution lists are delivered here and not to each member's mailbox. The ZOTnet then provides special software to allow these areas to be queried, read, and acted upon. In addition, private BBoards (ones in which only members are allowed access) are supported.

At present, this re–distribution mechanism is handled by a special channel on the gateway and the gateway's forwarding table. This is due primarily to an unfortunate technical problem with the way hooks are invoked on the gateway. So, let us see how the ZOTnet implementation attempts to overcome this difficulty. Items directed for distribution lists are aliased to a pseudo–host. This host is served by a special channel. The channel takes the message and, since it has the message envelope available to it, can take the proper actions with the message. This is precisely the reason a hook could not be used; the hook is given access to message body and not the message envelope. In order for the hook to distribute the message to the proper area, it would have to carefully examine the message header and try to deduce the address in the envelope that caused its invocation. Since the contents of the envelope need have no relation to the headers (especially after several aliasings), this is essentially impossible. The BBoards channel, since it does have access to the envelope, can make the correct decisions as to how the message should be re–distributed within the ZOTnet.

The idea that a channel should exist to perform some special processing not common to all MSAs in the MTS introduces an interesting topic for research. Distribution lists that reach large groups of users should conceptually have less priority in the MTS than messages between two users. That is, conceptually, distribution lists should be treated as bulk mail. In practice, distribution lists receive the exact same priority in using the bandwidth. This results in a large slowdown for other types of messages. The area of research that this suggests is some way to define different levels of quality of service in the MTS, and to have these enforced. In particular, research in the area of operating systems, which deal with how different classes of processor users should share the CPU might prove useful. Since these could be implemented on an MSA by MSA level and need not be implemented for the entire MTS, studying different interconnections of MSA following these policies to varying degrees might prove interesting as well.

Archive Handling

For users of the ZOTnet who maintain large records of computer mail correspondence, it often becomes desirable to be able to store these messages on a cheaper media, such as magnetic tape. In addition, discussion groups tend to acquire a tremendous volume of messages, all of which can not be kept on–line, and hence are often archived to tape. The problems of deciding what should be archived is not of interest to us. This policy would most likely be dictated by the amount of free space each system has available, and the amount of disorder that users are willing to put up with. What is of interest is how these messages, once archived, are retrieved.

It is important to recall that although messages are represented in text files, they are not just text. Messages have a semantics that transcends any semantics other text files have. In particular, messages have a header portion and a text portion, and user agents are able to act upon messages that adhere to the proper message format standards. This introduces one area of interest known as the message archive retrieval problem. That is, how can one best identify a message, using its message attributes as the search key. Further, once the message(s) has been identified, how can the message(s) best be returned to the user querying the archives?

The ZOTnet currently has no support for this area. Maintaining archives is a large problem, and as a result, this research topic has been introduced. Access techniques for off–line databases should prove to have some merit in dealing with th? first concern. For the second, the answer is obvious: have the archive server mail the message to the user!

High Bandwidth Transfers

The problem of having a 1200 baud channel between machines in the ZOTnet is not a concern for the majority of the ZOTnet's operations. It is only when message traffic increases heavily that the ZOTnet's capabilities become strained. This introduces the problem as to how the bandwidth can be augmented temporarily to handle large traffic loads, and then later reduced. Obviously, the augmentation must be expensive to maintain for an extended period of time, otherwise the ZOTnet would remain continuously augmented.

Recall that communications within the ZOTnet currently occur over terminal lines. In order to augment the ZOTnet's capabilities temporarily, it is believed that going out of band will provide a solution. That is, instead of using terminal lines, another media which is much faster and more expensive could be used. This question remains completely open, as no other form of media seems to present a reasonable (and inexpensive) interface to the two 2020 systems.

Conclusions

Although this paper has focused on several issues, there has been one central theme: interconnection can be accomplished cheaply. Computer mail which provides a powerful communications channel for users in a research environment can be extended from the local level to the local network level, and this extension can be done in an inexpensive fashion. In addition, the interoperability this provides is very valuable. Other utilities can be built on top of computer mail, and these can also prove to be very useful.

Further, this form of interconnection introduces many new and exciting problems. Some have been presented in this paper, and some solutions have been presented. Issues which have not been considered in this paper tend to be somewhat political in nature. These address access rights, divisions of responsibility, etiquette, and the concern over what level of cohesion the mail system should have with the local system operations.

Impact on the user community served in the ZOTnet domain has been both significant and encouraging. As any implementor would suspect, problems still exist in the operation of the ZOTnet, but as experience increases, reliability and utility has increased as well. The ZOTnet is providing the ICS research community with capabilities not previously available in its research environment.

Acknowledgements

From the standpoint of philosophy and rationale, the conversations occurring in two computer mail discussion groups had significant impact: Header–People at MIT–MC, moderated by David A. Moon, and MsgGroup at BRL, moderated by Einar Stefferud. From the standpoint of implementation, the ZOTnet represents a collection of different software systems supplied by different sources. Although these sources are quite numerous, special thanks should go to Jon Solomon of USC–ECL, and Professor David Farber of the University of Delaware.

References

[CSNET-DN1] L. Landweber, and M. Solomon, "Use of Multiple Networks in CSnet", Proceedings of the Compcon Spring 1982, (November, 1981). Also CSnet Design Note DN-1.

[CSNET-DN2] M. Solomon, L. Landweber, D. Neugengen, "The Design of the CSnet Name Server", CSnet Design Note DN-2, (November, 1981).

[CSNET-DN3] D.H. Crocker, E.S. Szurkowski, and D. Farber, "An Internetwork Memo Distribution Capability — MMDF", Proceedings of the Sixth Data Communications Symposium, (September, 1979). Also CSnet Design Note DN-3.

[CSNET-DN4] E.S. Szurkowski, "MMDF Dial-Up Link Protocol", CSnet Design Note DN-4, (April, 1980).

[CSNET1] L. Landweber, "CSnet - The Computer Science Research Network", Proposal to the NSF, (September, 1980).

[CSNET2] C.W. Kern, "CSnet Project Outline", NSF, (March, 1981).

[IEN048] Vinton G. Cerf, "The Catenet Model for Internetworking", IEN 48, (July, 1978).

[ISO97-16] ISO TC97/SC16 Document Number 227, "Reference Model of Open Systems Interconnection", (June, 1979).

[NBS81-5] D. Deutsch, F. Ulmer, J. Walker, "Features of a Message Transfer Protocol", NBS Report ICST/CBOS-81-5, U.S. Department of Commerce/National Bureau of Standards, (November, 1981).

[NBS81-?] National Bureau of Standards, "Specification of Message Format for Computer Based Message Systems", NBS Report ICST/CBOS-81, National Bureau of Standards, (September, 1981).

[NBS82-3] D. Deutsch, F. Ulmer, J. Walker, "Service Specification of a Message Transfer Protocol", NBS Report ICST/CBOS-82-3, National Bureau of Standards, (February, 1982)

[NMA-1] E. Stefferud, and J. McHugh, "The Role of Computer Mail in Office Automation", Computer Message Systems, Proceedings of the IFIP TC-6 Internation Symposium on Computer Message Systems, (April, 1981).

[NMA-2] E. Stefferud, "Economic Background for Computer Mail", Digest of papers, COMPCON-80, 20th IEEE Computer Society International Conference, (February, 1980).

[OSI-1] H. Zimmermann, "OSI Reference Model — The ISO Model of Architecture for Open Systems Interconnection", IEEE Transactions on Communications, Volume COM-28, Number 4, (April, 1980).

[RFC561] A.K. Bhushan, K.T. Pogran, R.S. Tomlinson, and J.E. White, "Standardizing Network Mail Headers", RFC 561, (September, 1973).

[RFC680] T.H. Myer, and D.A. Henderson, Jr., "Message Transmission Protocol", RFC 680, (April, 1975).

[RFC724] K.T. Pogran, John Vittal, D.H. Crocker, and D.A. Henderson, Jr., "Proposed Official Standard for the Format of ARPA Network Messages", RFC 724, (May, 1977).

[RFC733] D.H. Crocker, J.J. Vital, K.T. Pogran, and D.A. Henderson, Jr., "Standard for the Format of ARPAnetwork Text Messages", RFC 733, Internet Protocol Transition Workbook, (November, 1977).

[RFC791] Information Sciences Institute, "Internet Protocol", RFC 791, (September, 1981).

[RFC793] Information Sciences Institute, "Transmission Control Protocol", RFC 793, (September, 1981).

[RFC821] Jonathon B. Postel, "Simple Mail Transfer Protocol", RFC 821, (August, 1982).

[RFC822] D.H. Crocker, "Standard for the Format of ARPA Internet Text Messages", RFC 822, (August, 1982).

[X-MHS1] International Telegraph and Telephone Consultive Committee, "Message Handling Systems: System Model-Service Elements", CCITT Draft Recommendation X.MHS1, Question 5/VII, (September, 1982).

[X-MHS2] International Telegraph and Telephone Consultive Committee, "Message Handling Systems: Message Transfer Layer", CCITT Draft Recommendation X.MHS2, Question 5/VII, (September, 1982).

APPENDIX A
Abbreviations

The following (obscure) abbreviations are used (somewhat) consistently in this paper:

4.1bsd: version 4.1 of the U.C. Berkeley distribution of the UNIX operating system

CBMS: Computer Based Message System

FTP: File Transfer Protocol

IEN: Internet Experiment Notebook

IFIP: International Federation for Information Processing

IMP: Interface Message Processor

IP: Internet Protocol

ISO: International Organization for Standardization

MMDF: Multi-Channel Memo Distribution Facility

MSA: Message Service Agent

MTA: Message Transfer Agent

MTS: Message Transfer System

NIC: Network Information Center

OSI: Open Systems Interconnection

PSTN: Public-Switched Telephone Network

RFC: Request for Comments

SMTP: Simple Mail Transfer Protocol

TCP: Transmission Control Protocol

UA: User Agent