M.I.T. Laboratory for Computer Science

Request for Comments No. 238
January 17, 1983

Naming, Conversations, and Federation

by Karen R. Sollins and David P. Reed

Attached is the paper that we have submitted for inclusion in the Ninth Symposium on Operating Systems.

Naming, Conversations, and Federation

Karen R. Sollins and David P. Reed

Laboratory for Computer Science
Massachusetts Institute of Technology
545 Technology Square
Cambridge, Massachusetts 02139

January 17, 1983

Naming, Conversations, and Federation

Abstract

This paper considers a collection of naming problems that occur in electronic message systems when managing the names of mail senders and recipients. Using human communication as a basis, we identify five principles for naming. Human communication needs also help to identify several problems in naming. A solution is then proposed. We then highlight that the same naming problems occur in many other common situations in operating systems and subsystems, and that the same solution would satisfy these other situations. Therefore we conclude that a single, unified naming facility would suffice.

1. Introduction

Naming in computer systems spans a wide range of problems ranging from limitations of the underlying organization of the system to understanding the needs placed on the system by external forces. The set of system constraints that we wish to consider in this paper are related to the terms *autonomy* and *federation*. These terms will be discussed in detail. We will concentrate our efforts in the area of the external requirements on naming, and how they might be addressed.

The direction in which computer systems have been growing has been toward a multiplicity of machines interconnected by networks to provide a communication medium. The concerns of privacy and independence from other users have always been issues among computer administrators and users, but the nature of those concerns have changed somewhat as smaller cheaper computers have become available. In many cases, administrators purchase such computers and put them into service in isolation. At some later time, the administrators decide to connect the computers under their management. From here, the collection may continue to grow with little control and little consensus among the participants in such a "system". An *autonomous* computer is one for which all decisions are made independently of the decisions made for any other; all the activities on one computer are isolated from the activities of any other. This has been the first direction that many administrators have gone in order to escape large time-sharing systems. We model the move to interconnections using a network or set of networks as a move toward *federation*. In a federation, there is some agreement on behavior and protocols to be utilized, but the barriers apparent in the isolated machine are still available to anyone who wants to enforce them. If the administrator or user wants to disconnect the computer from the network by simply not accepting messages, that is possible. If that computer provides a service to the participants in the network, they must understand that such a service will not always be available. On the other hand, as long as administrators and users want the communication available, there is common ground on which it can occur (such as agreement about protocols and services to be available). It is this loose coupling that we label federation, and it is our assumption that this is the underlying model of our system.

The ultimate goal of a system should be to provide a useful tool for the users. Therefore, in Section 2 we will focus on the human requirements in order to pinpoint the technical issues involved in providing a naming facility. As a vehicle for addressing the human factors, we will consider the problems that arise in naming recipients using aa electronic message system in Section 3. We will then address a solution in Section 4, and finally, in Section 5 will consider other problems, pointing out their similarities to the problems presented in Section 3. Section 6 summarizes the paper.

2. Human naming requirements

In order to bring our concerns into focus, we will consider a particular set of issues that occurs in the human to computer interface. The context we will use is that of an electronic message system, to allow users to send messages to other users at other computers in the federation. In particular we will consider that names that might be used in the "reply-to" field of a message from user A to user B. In this case the names need not only to provide information that may be used by computers, but also names that have meaning to the users involved. When A chooses the names for the reply-to field, she will choose the names by which she knows people. When B receives the message, as much as possible he would like to see the names by which he knows that same set of people. In order to clarify our discussion, we have identified a set of goals for such a system[1]:

1. **Multiplicity of Names:**

   - *The system should allow different people to use the same name for different things.*

   - *The system should allow different people to use different names for the same thing.*

   - *The system should allow a single user to use different names for the same thing and the same name for different things in different contexts or at different times.*

   In naming the entities in their worlds people often choose names in overlapping sets. When A and B say, "my desk," they mean different desks. On the other hand, when A talks about her own desk she will say "my desk," while when B talks about it he will say, "A's desk." If A has a desk at home and a desk at the office, when she is at the office she will speak of the desk there as "my desk" and the desk at home as "my home desk" while

[1]Lindsay [4] has identified a similar set of goals for the catalog of the R* system, but because his goals are somewhat different, his solution is also.

when at home the home desk as labelled as "my desk" and the desk in the office is "my office desk." This sort of thinking has led to the principle of providing a multiplicity of names.

2. **Locality of names:** *A person should be able to have contexts to reflect his or her focus of interest. These contexts may be shared by several users or used by only one. A user should also be able to use two or more contexts to reflect a focus between or including several contexts.*

   Again, we will consider the example of naming desks. A has two contexts, one for home and one for the office. Now, we will add to those contexts the name C. In the context of home, C is A's roommate, whereas C at the office is another co-worker. When A and B mention the name C while at the office, they both agree that they are in the context of the office and that C is their co-worker. When they mention A while in A's home, they again agree, but, in this case, that C is A's roommate. It is certainly possible that A might wish to say something about roommate C to B while they are at the office. In this case, although the primary context may continue to be the office, they will also temporarily be placing themselves in the home context, at least for the resolution of the name C. Once the contexts of been clarified to both A's and B's satisfaction neither would want to be required to identify C uniquely among all people with the name C. It is the need for overlapping, multiple, and shared contexts providing unique identification among only a small number of entities that led to the principle of locality.

3. **Manifest nature of names:** *Names have meaning to humans outside the computer. Therefore the human must have control of naming; it should not be controlled by the needs of the computer. Computers and humans must create names meaningful to other humans. Names must be transportable while outside the system.*

   We will now consider the situation in which A creates a file on the computer that she wants B to read. She uses an editor and must use a name for the file that the computer can handle. She also wants a name for it that she can remember, and hopefully will have some meaning to her. What is more she then wants to be able to walk down the hall and tell B the name of the file, so that he can then use that name to look at the contents of the file. Hopefully the name will have some meaning to B also, so that he will be able to remember it too. Therefore, since it is unlikely that the computer would choose a name that has much meaning to either A or B and since the name must be useable outside the computer system, we are led to the principle of a manifest nature of names.

4. **Flexibility of usage of names:** *Different sorts of names should be allowable, reflecting human patterns of naming. For example, descriptions should be available. The user ought to be able to attach them to objects. Full and partial descriptions ought to be available. Users ought to be able to use generic names to label classes of objects. These generic names may be labels or descriptions. The response to use of a generic name is a set of objects. What use is made of the set is up to the user or program using the generic name. The naming facility should not impose a selection function on generic names. In addition to descriptive and generic naming, the human should be able to use combinations of names in order to narrow the set of objects that are named.*

   Now, we will consider what B might do with the file he was asked to read. If he prefers reading a printed page he will want to send it to a printer. Perhaps there is a service on the system that manages all the printers. His first choice will be "The printer on my floor" (with the implication of it being in his building as well). He has no interest in remembering a name for it; a description is just what he wants. When he discovers that that printer is being repaired, he decides that any printer will suffice, because he knows it will be delivered to his office in interdepartmental mail. For this he uses the name "printer" to mean anything in the class of printers. This is what we call a generic name. In fact, descriptive and generic naming can be combined. It also may be the case that use of the printer on the third floor is restricted, so B may want to use the logical combination of "printer" and "not on the third floor". It was these sorts of flexibility requirements that led us to this principle.

5. **Usability of names:** *Humans ought to be able to manage names in the computer system as easily as they do without computers. This means that*

   - *changing, modifying, or adding names should be easy.*

   - *moving among contexts should be easy.*

   - *global knowledge should not be necessary in order to create or use names.*

   - *it should be easy for the human to disambiguate names so as to allow for authentication, thus avoiding accidental ambiguities.*

   Now, we will consider what might happen if A's roommate C joins the company also, and A wants to send a memo to her on the computer. A will want to add C to her context in the office. There is no reason to require that each name must be unique and in fact in order to provide generic names we cannot. A should be able to incorporate two people under one name in her office context. This action should be as easy to accomplish as learning the names of two people and remembering them. It should also be easy, for instance, in discussing desks with B to identify whether the context is office or home and to switch between them. It certainly should not be necessary for A to know whether there are other people within the company with the name C, in order to use that name locally. Finally, if A wants to send a memo to C her roommate, she will need help in disambiguating the name, and she ought to get that help. Basically, it should be easy for A to understand and manage her names and move among her contexts. If this is not true, the naming facility will not be used.

There is a requirement that, in the past, has been placed on many systems, globally unique names or identifiers (*uids*) underlying a naming facility, in some cases even directly apparent to the users. There are two reasons for providing and using uids;comparison and identification. First, an object will have exactly one uid which is guaranteed to be unique in the universe in question. Therefore, given two names, a simple comparison of the uids into which the names map will indicate whether they are naming the same object or not. Second, since an object will have exactly one uid for its lifetime, and uids are guaranteed to be unique for all time, it should not be difficult to ensure access to the same object at successive times. Again, this can be achieved by a comparison. In centralized systems, it was easy to guarantee that names were unique since they could be checked centrally. As computing nodes became distributed and autonomous, the ability and need to check that names were unique across machine boundaries disappeared. With the growth of federations of computers, many system designers have returned to the assumption that globally unique names must be available and have proceeded to design systems on this assumption. There are two reasons that globally unique names should not be required: first, globally unique names are difficult and sometimes impossible to guarantee; and, second, it is not clear that they are needed.

In order to be able to guarantee that names are globally unique, there must be either some means of checking them after they have been created or some means of guaranteeing during generation of them that they are unique. The difficulty arises when a computer changes from being autonomous to being a member of a federation. If one follows the route of attempting to generate only unique names and at some point a new computer or network joins another pre-existing network, there must be some reconciliation in order to guarantee global uniqueness of the names already in use. From that time forward there also must be a new agreement guaranteeing that all new names will be globally unique. Both of these stages are difficult at best, and may be impossible if the members of the federation cannot agree either on the reconciliation or on the new procedure to follow reconciliation.

Even if guaranteeing globally unique names is not unattainable, it is not necessary. If globally unique naming were required at some level in the system, it should not be imposed on the users as implied by principles multiplicity, locality, and useability of names. What is more, relative naming can certainly provide the basis for the lowest level naming.[2] In fact this is what is currently used at the network level in many cases. Consider, for example, a network in which all nodes have assigned to them globally unique names and the network is not a broadcast network. Regardless of how the route is determined at each forwarding point, that node is choosing a wire on which to send the outgoing packet by translating the destination into its own local name for that wire. If the network is a broadcast network, nodes will need names that are unique within their own broadcast range. It is still the case that if two such broadcast networks are connected by a bridge of some sort that naming can be relative to one or the other of the networks. To name a node in the other network, one would name the bridge and some name that the bridge can translate into the appropriate name that is unique to the second network. Even in the new proposals for modification to naming on the ARPANET ([6]), where a claim is made that naming should be by hierarchical, globally unique identifiers, imposed by a central authority, an escape mechanism is provided. This proposal recommends what are called *domains*. In the past, a person's address on the ARPANET was composed of a user name and a host

[2]Source routing as discussed by Farber ([2]), Sunshine ([7]), and Saltzer et al. ([5]) provide us with another example of relative naming carried into places where previously it was believed that global naming was required. This will be discussed further in Section 5.

name. The new proposal suggests that the host name be replaced by a domain which is a hierarchically structured name representing nested management authorities. The escape mechanism provided is that there be domains within which the proposed naming conventions need not be enforced. This escape mechanism was developed to accommodate those networks whose administrators would not or could not agree with the ARPANET standards. Here we have clear-cut evidence of federation occurring and globally unique identifiers falling victim to federation. Since globally unique naming is not feasible at the low levels and should not be visible to users or their programs, we will not assume that it is available.

3. Example: Electronic message systems

Electronic message systems provide a good example of the naming requirements. Here we consider the problem of assigning and using names for recipients of electronic messages.

Inevitably, electronic message systems developed by individual organizations will be interconnected with those of other organizations. This is a clear case of federation. Each organization is autonomous, so name assignment and resolution necessarily must be decentralized.

Decentralization of name assignment and resolution arises from lack of global authority and needs for local freedom. No one authority can know or control the extent of interconnection of electronic message systems. Thus, multiple autonomous authorities must exist, and further, each authority must operate without full knowledge of the extent of the network. In addition, the act of naming is fundamental to human communication. Thus, individual participants in the system have a strong need to be able to create new names, especially nicknames and abbreviations, in order to make the names they use more meaningful to themselves and those with whom they communicate.

The problems encountered with the use of names for originators and recipients of electronic messages can best be illustrated by a scenario of typical message activities. It should be quite apparent from the following that hard naming problems result from the linkage between the human use of names and the automatic resolution of names provided by the message system.

The actors in our play are Tracy Antiope, head of the International Network Folk Song Tablature Protocol Standards Committee (called by its members INFSTPSC or "the committee"), and Randy Link, a newly appointed representative to the committee from a small but important music publishing company (or to be more precise, an "electronic publisher" that distributes its material on the world network rather than on paper).

To name Randy to the committee, Randy's boss first sends a message (Figure 3-1) to the head of

INFSTPSC, telling Tracy of Randy's new status as a representative. Tracy responds by sending a message (Figure 3-2) to Randy with "electronic copies" to all members of the committee, thanking Randy for serving and introducing Randy to the members of the committee. Randy later sends a message (Figure 3-3) to Tracy, suggesting that INFSTPSC form a joint working group on Zither Tablature with the International Zither Players Organization (IZPO, or "the organization," of which Randy is the chair). Tracy endorses the idea, and nominates several committee members to join the Joint Zither Tablature Working Group.

Figure 3-1: Message to Head of INFSTPSC

```
From: J. O'Connell, President, Chord Networks
To:   Chairperson, INFSTPSC

Randy Link of our company will be our representative to INFSTPSC....
```

Figure 3-2: Message to Members of INFSTPSC

```
From: Chairperson, INFSTPSC
To: Randy Link, and members of INFSTPSC

Welcome, Randy, to INFSTPSC....
```

Figure 3-3: Message to Head of INFSTPSC

```
From: Randy
To: Chair, INFSTPSC
Subject: Proposal for joint working group with IZPO

lksjdflkajsdfsalkjdfsaldkjf
```

It is easy to elaborate this scenario, but we will stop here, since this short scenario has exposed a plethora of naming requirements. Harking back to our enumeration of goals for a naming system, we find that the scenario illustrates every point quite well.

At the end of the scenario, Randy plays a role in four organizations (Randy's employer, the committee, the organization, and the working group). They are linked, but no single authority controls the structure of the entire group. Hence, we have a clear illustration of how federation of autonomous groups will occur.

- *Multiplicity of names* arises in several ways. In the scenario, Tracy has multiple names, one name that represents Tracy's committee role (INFSTPSC-Chair), and a name that represents Tracy as a person. Just as in ordinary mail, it is necessary to allow mail to be addressed to a "role", so that one can allow for personnel changes by rebinding a new person to the role name. When Tracy sends messages (especially messages outside the committee), it is important to indicate which of Tracy's roles the message serves (chair or private person), so that eventual replies will be directed to the right person if Tracy is replaced as chair.

- A member of the working group might refer to "the chair" in a message sent to the working group without ambiguity. If the same message were to be sent outside the working group, the reference would have to be qualified with the context. *Locality of names* allows for development of working contexts where elaborate qualification is not necessary. But see below for a discussion of the "reply-to" bug that can result from handling this locality improperly.

- The example of using "the chair" within a group illustrates our point about *name meaning being manifest*. In the context of the message, which the reader can derive by a thought process that depends on the name of the source and the name by which the recipient is addressed, the meaning of "the chair" is clear. But if the source name or the destination name is ambiguous, then names in the message are likely to be ambiguous. For example, names in a message from "the chair" to "the group" will be hard to understand for members of the group who are members of several groups. But note that removing ambiguity does not guarantee understandability -- were we to name people by their social security numbers (or a more global unique name), recipients would not find it easy to understand the relationships of names to the people with whom they work (digitized pictures would be good unique ID's for non-blind recipients). Similarly, senders ought to be able to generate addressee names from knowledge they have outside the system. (A prime example of this problem is that Randy's ARPANET address might be "RQL at INFSTPSC-FROB". Since Randy might have accounts on several timeshared machines, and own a personal computer, someone would have little idea of how to address an ARPANET message intended for Randy).

- The use of names for roles and for groups of people illustrates *flexibility in the usage of names*. Role names are a case of descriptive names, and group names are an example of generic names.

- There are many concerns that surround the *usability of names* in the scenario. A new working group is created and must be given a name quickly. Names provide a basis for authentication, as when Randy's boss sends a message introducing Randy to Tracy.

The naming mechanism used in the ARPANET for electronic messages provides examples of a number of problems that result from a poorly thought out naming framework [1]. Each message sent contains a header that contains names for recipients and a name for the sender. The names are of the form "<name> at <host>", where <name> is a character string interpreted by the target machine, and <host> is a character string that specifies the target machine.

There are two primary problem areas with the standard. First, the context in which names are to be resolved is often not clear. Second, and more important, the relationship of the names to names used externally is often difficult to determine.

The focus of the standard is specification of a mail transport mechanism, so the names are designed to route messages from the original sender to the ultimate receiver, without ambiguity. Thus, when the names are entered into messages by the sender, the sending computer and any intermediate forwarding computers must be able to determine where to ship the message next. Multiple destinations are allowed, so mail can be sent to "Randy at QRYX, [1234,5678] at Podunk-KA". Some sites also maintain mailing lists that allow a single name in an incoming message to be interpreted as a generic name, so the message is forwarded to multiple targets. The "at <host>" field may be omitted, when the mail is targeted for the sending host.

Since names are intended for routing, they may not be meaningful outside the context in which the sender originally sent the message. The best example of this is the "reply-to" problem. When a recipient receives a message, he/she may want to generate a reply to all of the original recipients of the message plus the sender. The "reply-to" problem results when the names placed on the message by the sender do not mean the same thing to the recipient's system. For example, when Tracy sends the message in Figure 3-2, the names of the members of the committee are meaningful to Tracy and the systems involved in forwarding the various copies. When Randy receives a copy, the names of the other recipients may or may not be usable as mailing addresses from Randy's point of view. Thus, Randy's message system might not be able properly to reply to the message, because the address would have no meaning or the wrong meaning in Randy's system.

The ARPANET has partially solved this problem by requiring that mail systems use globally unique host names chosen out of a single global context (i.e. the Network Information Center or NIC maintains a list of official names for all hosts that can be reached in the ARPANET). However the problem still exists in the case where the sender omits the "at <host>" field, or where a mailing list is specified. Some message systems on the ARPANET attempt to fix up the case where a name is specified without the "at <host>" field, by adding the name of the host from which the message is sent. This heuristic does not work if the message has been forwarded, or if the sending host has implemented an abbreviation facility that allows a name not qualified by an "at <host>" field to be used to route a message to a different host.

The requirement of globally unique host names does not allow for federation of independently constructed networks. In construction of a federated network, global uniqueness of host names cannot be achieved without imposing the inconvenience of changing the names of hosts, even though some users might not even be aware that the interconnection is happening.

Parallel to the "reply-to" problem is the "name-equality" problem. When Tracy receives a message, it may be quite difficult to tell who "chris at INFSTPSC-FROB" really is. It might actually be the same person that Tracy calls "Smith at INFSTPSC-FROB", since the electronic message system at INFSTPSC-FROB may implement multiple names for the convenience of its users. The "name-equality" problem is one of whether or not one can equate two names for the same entity inside the computer system, especially in our federated system. It is this this authentication of names that can be a problem.

Related to the "name-equality" problem is the "who-is" problem that results from the fact that a user may use different names outside and within the computer system Randy may use the initials RQL as a name inside the system, but there is no guarantee that the name "Randy" will be associated with "RQL". If this is true, there is no way to ask the local host if RQL and Randy are the same person.

Finally, in the ARPANET message standard there is no way to have a name that is not fixed to a site. The name (title) INFSTPSC-Chair might be attached to people from a variety of organizations (hence probably distinct sites) at different times. It is possible that there is no single site that is willing to maintain the binding between that title and the current person that corresponds to that title by forwarding all the mail intended for the chair. Instead, a desirable solution not implementable in the ARPANET is a name that has a translation that may move from site to site over time. Such dynamics are important even for names that are not titles, too. For example, Randy might change employers while still remaining a member of the working group. A change of employers might result in relocation of Randy's mailkbox to a new site. Randy's name within the working group should not be changed gratuitously just because of a change of employment. We will call this problem of dynamics the "mobile-name" problem.

4. The solution

In this section we will present a mechanism for solving the problems presented in Section 3. We will then discuss how the mechanism can be used to solve the problems. Section 5 will then discuss more widespread problems and outline briefly how our mechanism might be used to solve them.

The basis for what we are proposing is a simple, non-restrictive type of object called a *context*. A context translates names into entities from the point of view of the user of the context. A name is a label that allows the user of that label to refer to the entity by the label of his choice. In some cases, a name will be translated into another name less meaningful to or less easily used by the user of the original name. Further context translation may then be requested. In the remaining cases, the user or program will use the resulting translation as is. The result may be the full name and address of a recipient, the first hop in a route to the recipient of a piece of mail, or one of many other possibilities. Whether further translation is needed or not, the decision is not made within the context but by the user or program requesting the translation.

In addition to contexts, we propose another mechanism, *conversations*. The reader should be aware that although *conversations* mechanism is based on our example of human communications, it will have a more general use later in the paper. A conversation has two parts, the *current context* and the *environment*. When two people communicate, there is a small set of names that they use regularly and to which they may add new names needed in that conversation; it is this current context that they share. They each also have a large pool contexts from which to draw names into the current context. These pools may be different for each participant in the conversation. The pools, which we call their environments, consist of collections of contexts, which may or may not be partially ordered, but which are needed to translate names which may not be in the current context. The current context is shared by the participants. Other contexts may also be shared. A context that is the current context of one conversation may be one of the contexts in the environment of another conversation. In order to understand contexts and

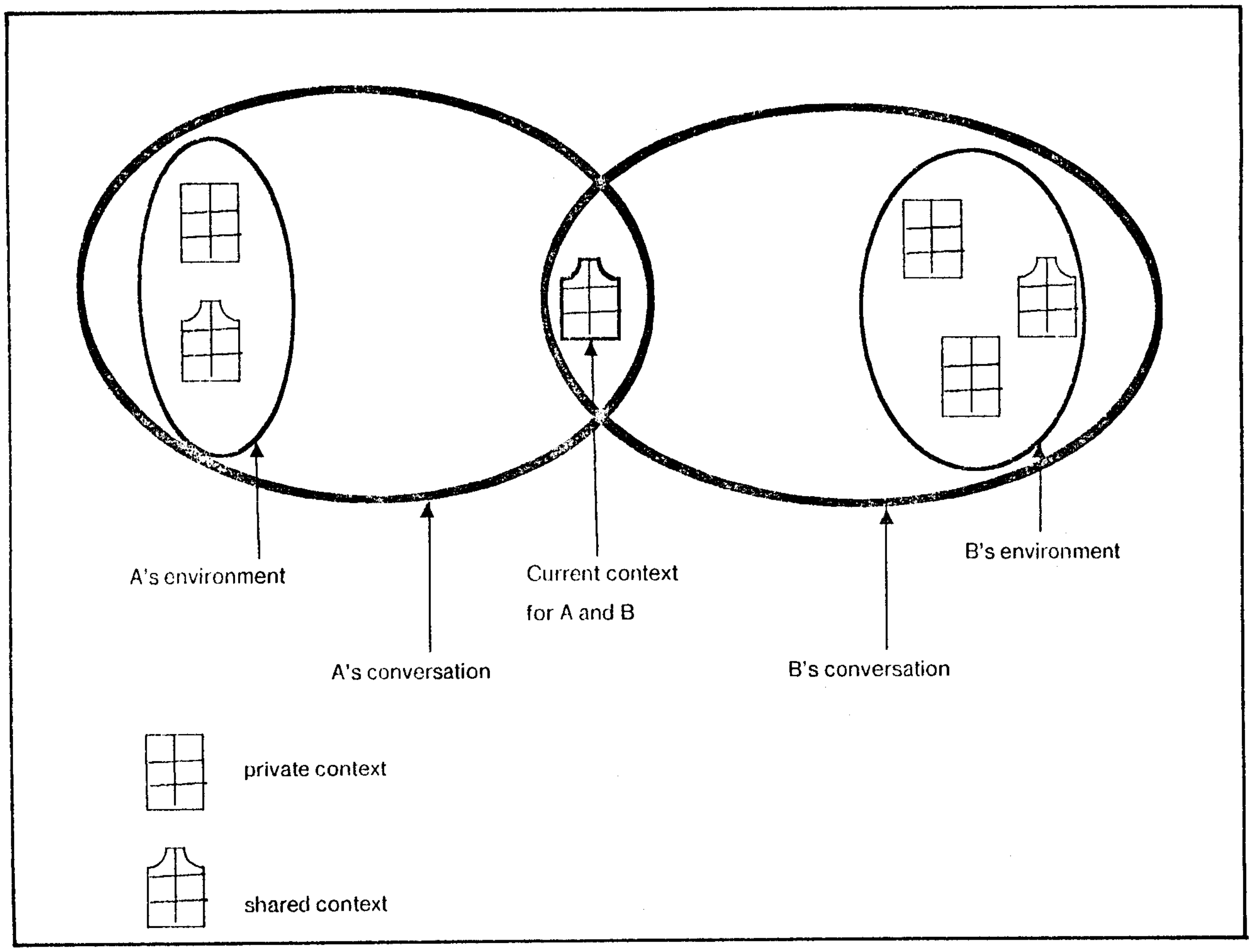

Figure 4-1: Two points of view in a conversation

environments better, Figure 4-1 depicts two views of a conversation, A's view and B's view. In order to converse, A and B must share a current context. As with human communication, each has his or her own set of additional contexts, his or her own environment to use in the conversation. A has one private context and one that is shared with someone, perhaps B, but perhaps not. B has two private contexts and one shared context. In this figure, the contexts in the environments have not been given any ordering, although in practice they may be given a partial or even a complete ordering. The shared contexts may be current contexts of other conversations or not. There is nothing different in the handling of shared and private contexts when they are in the environment of a conversation. It should be remembered that in any conversation some context must refer to another conversation in order to escape that conversation. Contexts and conversations are types of objects, and as such have operations defined on them. Tables 4-1 and 4-2 describe the operations on contexts of importance here (they have been divided into two tables for ease of reading), and Table 4-3 describe those of importance for conversations.

Table 4-1: Operations on Contexts

create	creates a new empty context and assigns it to the name passed as an argument in the current context.
append-context2context	appends the translations in the first context to the second context.
destroy	destroys the context specified.
current-context	returns any names the current context knows for itself.
create-image-context	creates a new context named in the current context, containing the complete translations for all the names in the current context.

Table 4-2: Operations on names in contexts[3]

add-name	add a translation pair to the specified context.
delete-name	delete a translation pair from the specified context.
translate	translate a name in the specified context into an object.
equal	returns True if the two names name the same thing, otherwise False. Each name has associated with it a specified context.
names	is an iterator that returns, one at a time, all the objects named by the specified name in the specified context.
how-many-translations	returns the number of translations for the given name in the specified context.

[3]Each of these operations has a default form that assumes the specified context to be the current context. It also should be noted that these operations are operations on contexts, but have been separated into two tables for clarity.

Table 4-3: Operations on Conversations

create	creates a new conversation with a new empty current context and empty environment assigning the new conversation the given name in the specified context.
create-with-shared-current-context	creates a new conversation with the current context set to the context specified, assigning the new conversation the given name in the specified context.
current-context	returns the current context.
add-context	adds a context to the environment.
delete-context	deletes a context from the environment.
order-environment	allows the client to order, or partially order the contexts in the environment of the specified conversation.
change-conversation	switches to the specified environment.

Now let us return to the scenario presented in previous section. As in that section, and for simplicity, we will consider only the problem of managing the names of people. Initially, each member of INFSTPSC will have a conversation for that committee, and all will share a current context. That context will include the names of all the members. It will also include an entry for "the chair". Randy and the other employees of the publishing company will have another set of conversations, sharing a different current context, that will contain the employees' names by which they are known at the publishing company in general. In addition, Randy may have a private context that includes private nicknames for the people in adjacent offices. This context may be included in Randy's conversation at the publishing company. A similar situation holds in IZPO, except that in this case, "Randy" and "the chair" will be translated into the same entity, at least at present. (Randy may not retain the chair, with all the new responsibilities.) When Tracy accepts Randy into INFSTPSC, Randy will be added to the current context for INFSTPSC. As acknowledgment of the acceptance, Randy will be sent some representation of the current context, so that it can be included in Randy's view of the conversation. The current contexts of the IZPO and the publishing company conversations will need to be included in the environment of Randy's new view of the INFSTPSC conversation. To do this Randy will first invoke *create-with-shared-current-context* passing it the current context sent by Tracy and a name (probably "INFSTPSC") in the current context of the conversation in which Randy is running at the time. Then, *add-context* will be invoked to add contexts to the new environment. Randy must be sure that somewhere in that new conversation there is a name for another conversation, so that escape is possible. Now Randy is ready to participate in INFSTPSC. It is possible that, in the current context of INFSTPSC, in addition to the names of the members, there are names of other interested people.

These groups will be distinguished by the labels "member" and "nonmember". When Tracy sends the message to the members, it will be to the generic name "member". Anyone answering to that name will receive a copy of the message announcing Randy as the new member. Now, finally when Randy suggests the joint working group, a completely new conversation will be needed. For this, Randy will invoke *create* to acquire a new conversation with an empty current context, reflecting the fact that nothing yet has happened. As members are appointed, they will create their own versions of the conversation, using Randy's new current context, into which Randy will be placing their names when they are appointed. Now Randy has four conversations available. It would be most convenient if each named all the others, in order that Randy can switch from one to another without having to say "The conversation I called IZPO at the publishing house" when switching from INFSTPSC to IZPO, but simply invoking *change-conversation* when a message arrives as part of the IZPO conversation.

We now have a complicated structure with shared current contexts being a focal point. How the shared contexts are managed will determine to what degree the problems identified in Section 3 are solved. Briefly the problems arise as follows:

- the "reply-to" problem arises if a name has different translations or meanings in different places and which translation is to be used is not specified completely enough.
- the "name-equality" problem arises when, inside the system, one has two names and one wants to know whether they refer to the same entity.
- the related "who-is" problem arises from trying to equate names external to the system with names internal to it.
- the "mobile-name" problem arises when a name does not change, but the entity into which it is translated not only changes, but moves from one location to another.

We will consider each of them separately.

First, let us consider the "reply-to" problem. Here we are concerned with a name having different translations in several different available contexts and not knowing which context to use. One situation in which this may arise occurs when the current context of a conversation is distributed at several sites and updates to the current context are not made atomically, or even promptly enough. This would suggest that in order to address this form of the problem atomic updates may be needed. At the same time, it should be possible for a part of the group involved in a conversation to proceed and update the shared context without all the participants being available. In order to provide for this, some version of Gifford's weighted voting ([3]) would suffice. Returning to the original problem, if it arises from a lack of shared contexts or a lack of information (names of contexts) being passed, there is little we can do, other than what humans do in conversation, which is to ask questions and begin further exploration of possibly shared contexts.

The "name-equality" problem, or discovering whether two names refer to the same object, is a more difficult problem. It is not always solved satisfactorily in human communication. The related "who-is" problem is even more difficult to solve because it involves human communication as well. Let us return to Randy. One of the other members of INFSTPSC might ask whether the Randy from the publishing house is the same person who chairs IZPO. Randy may have two mailboxes in order to keep the two subjects separate. If Randy were assigned some globally unique identifier that was attached to every reference to Randy, the "name-equality" problem could be solved by being able to equate the owners of the two mailboxes. This still would not solve the problem of equating the Randy one meets on the street with the owner of the mailboxes unless that Randy presented a uid. But as discussed in Sections 2 and 3 there are other problems with globally unique identifiers as well. They are difficult, and maybe impossible, to guarantee in a federation. They are inflexible. They are not useful to humans as names. What is needed in both cases is an authenticator that is as effective and reliable as what is used in human communication. Among people a variety of mechanisms are used. Some of these may be helpful in suggesting how this problem might be solved in computer systems. The simplest form of elucidation is to ask about other names (and descriptions) that the entity in question might have assigned to it, perhaps in another context. The person asking about Randy may be told who it was Randy that suggested the Joint Zither Tablature Working Group, and therefore it is likely to be the same Randy who chairs IZPO. It is the use of hints and pieces of description that is often most helpful. In identifying people, their looks are probably their most unique identifier. Even in this situation, we must be prepared for incomplete identification because there is a possibility of identical twins. Along these lines, we propose that entities be given identifiers that are as unique as possible, but with the understanding that they are not guaranteed to globally unique under all conditions, such as federation. These moderately unique authenticators should be passed out as an entity is assigned names in new contexts. By distributing them, we solve the problem that the site containing any particular entity may not be available, and is not needed in order to handle the question of whether two names identify the same entity. If two such authenticators are different, then we can be sure that the entities are different. It is only in the case where the authenticators are the same that there may be some doubt as to whether the same entity is being named twice. In the final analysis, the only way to guarantee that the answer to the question of equality is by having a guarantee that authentication provide an absolute and universally valid answer.

Finally, the "mobile-name" problem is addressed by contexts themselves. In this case, we are concerned about a translation changing while the name does not. The original principle of multiplicity of names addressed this problem, allowing a name to have different translation at different times. Contexts can do this; someone must simply change the translation using the *add-name* and *delete-name* operations.[4]

In this section we have presented a mechanism composed of contexts and conversations with their operations showing how they can be used to address the problems presented in Section 3 in light of our goals of Section 2. The next section will present briefly several other examples of situations in which similar problems arise.

5. Extensions

In general, in computer systems, there are separate naming mechanisms for the various kinds of entities that need to be named. The mechanism for naming people is different from the mechanism naming files which in turn is different for the mechanisms for naming processes, ports, and other entities managed by the system. For the most part, these latter are inaccessible to or at least difficult to access by the normal user. We will consider briefly three examples in addition to the electronic message example, naming text to be read and edited, naming a printer service, and source routing in a network, concluding that the problems and their solutions are similar to the electronic message situation.

Naming text and editing

We will consider another scenario with our same actors Tracy Antiope and Randy Link. The publishing company has been using its own zither tablature that has the advantage that it is easy for the publishers to produce. Meanwhile the members of the International Zither Players Organization (IZPO) have developed a different tablature that is easier for them to read as zither players. The Joint Zither Tablature Working Group (JZTWG) takes as its first task resolving the differences between these two. There is documentation on both tablature specifications available on computers, but on the computers of IZPO and the publishing company respectively. The INFSTPSC also has its own computer. The JZTWG is composed of members of the other three organizations, and the work of standardizing will be done jointly by members of all three organizations. Again, this simple scenario will highlight many problems.

First, let us consider Randy's work on the tablature specification. Initially, the ideas documented at the publishing company and IZPO must be accessed. They will not be modified, but Randy will read them. Then a new document that is part of the work of JZTWG must be created, using parts of the

[4]If one is particularly concerned, a *change-translation* operation might also be included to provide atomicity so that neither the "reply-to" nor the "name-equality" problem arises between the execution of the two operations, depending on the order in which they are executed.

two other documents. Since the JZTWG is not part of any one organization, all three have agreed to allow their computers to be used for the work of this new group. As with the electronic message situation, we can see in this situation all of the issues raised in Section 2.

- Within each of IZPO and the publishing company its own tablature documentation may be called "the tablature documentation". Because of this, Randy may want to name the tablature documentation from IZPO "the IZPO tablature documentation", while continuing to call the publishing company's "the tablature documentation" while within IZPO the name "the tablature documentation" will continue to be used. This need for a multiplicity of names was our first principle.

- The range of items needed for Randy's work on the tablature is centered in a new context defined by the JZTWG that includes some items from the worlds of both the publishing company and IZPO. This exemplifies the requirement of locality of names.

- The need for the meanings of names to be manifest can be seen in the fact that Randy needs to be able to name and find the two sets of tablature documentation and create a new one that other people will be able to find and recognize as such. Creating names for them that have meaning outside the systems suggests that other people can use them without having to be told what names to use.

- The flexibility requirement will help Randy in labelling the different documents on tablature. The IZPO one may have a descriptive name, indicating its source. The one produced at the publishing company may have a nickname such as "tabdoc" because that is easier to type and everyone there knows what "tabdoc" means. Finally, the editor Randy is using may allow the simultaneous creation of multiple windows for editing and references. Randy may want to create a generic name, "current tab documentation" which in turn refers to all three, the two already produced and the new one in the works, in order to give the editor this one name for the three documents.

- Finally, in order to create nicknames such as "tabdoc", generic names such as "current tab documentation", a new conversation such as the JZTWG, the operations on names, contexts, and conversations must be easy to do. If Randy has to think a great deal about it or collect names from many sources to achieve what can be achieved with a few questions and a minor discussion among the participants when talking face to face, another means for achieving the naming, possibly less elegant, but simpler to use, will be found. The usability of any facility is of primary importance.

A print server

Special printers are needed to print the tablature, not all printers are capable of printing the needed fonts; there is one at the publishing company and one at INFSTPSC, but none at IZPO. Randy does some work on the tablature specification in preparation for a meeting of the JZTWG, and prints a copy, but is unable to attend the meeting, and therefore requests that a copy be printed at INFSTPSC for Tracy to take to the meeting. At this point Randy wants to print two copies of the work done to date on the proposed standard. The print server will receive two requests for the same document, but with slightly different requirements, the first that the printing be done on the tablature printer at the publisher's office, and the second that the printing be done on the tablature printer at Tracy's office. Again, as with naming people and documents, each of the original requirements can be seen to come in to play in this situation. In order to avoid further repetition, they will not be discussed further here. Let it suffice to say, that a similar set of problems arise whether one is naming people, data objects, or other processes.

As with the electronic message situation, contexts and conversations can solve the problems raised in the editing and printing scenarios. Again we will make use of the same set of conversations, one for each of the organizations, INFSTPSC, the publishing company, IZPO, and the JZTWG. We may now have additional contexts in the environments, although that is not necessary. But Randy, starting from the publishing company conversation, must be able to name at least the JZTWG conversation, from which the other conversations should be nameable. For simplicity, the participants in the conversations may have divided their entities by categories, such as a context for people, and another for documents. They also may have chosen other ways to break apart the collection of entities, or they may simply have decided not to separate them. In any case, the ways in which names, contexts, and conversations would be handled in these scenarios are the same as in the electronic message situation. The problems and solutions bear a great similarity to each other.

Source routing

Lastly, we will consider source routing in a network protocol, as mentioned in the footnote on page 5 ([5], [7], and [2]). As described by Saltzer et al. source route generators provide approximately what our conversations provide. They may or may not be hierarchically arranged, but more importantly, they can operate independently and need not depend on any global naming scheme. The source routing mechanism meets all of our original five goals. Clearly both at the level of the names handed to the routing servers and at the level of the routes returned by the routing servers, a multiplicity of names will (or at least can) be used. There is no reason that routes will be unique unless for some reason every node or position on every local network is assigned a globally unique name. Fortunately, since this is difficult to do, it is not necessary, thus leading us to the possibility that the same route description may move packets between two completely different pairs of locations. Source routing makes no assumptions about how and where the source routes are generated, but instead allows for a combination of the local hosts and remote route servers. This can provide a degree of locality not provided by other routing schemes. The fact that name meanings need to be manifest is apparent because the names that are used for locations must be meaningful to the requestor and the routes that are returned must be intelligible to the protocol daemons attempting to deliver packets. In this case, we have gotten away from humans needing to understand and find the mechanism easy to use, but these requirements are still present for any client of the facility. In fact, as we will see shortly, there still are human requirements also, in that humans will be doing the programming and the facilities must be easy for the programmer to incorporate into programs. Because different kinds of protocols (mail, telnet, file transfer) may all be using the same mechanisms, a certain amount of flexibility must be available. Finally, as mentioned earlier, useability is important, particularly for the programmer, and as discussed by Saltzer et al., one of the benefits of source routing is that it allows for simple operation within a confederacy without imposing to any great extent on any individual member of the confederacy.

As we have seen in the examples presented in this section, the requirements for naming mechanisms are similar for situations which historically have been kept separate. We propose that a single mechanism such as contexts and conversations should be used for these and the other naming problems in systems. In fact, in addition to the simplicity gained by providing only one naming facility, there is another benefit. Names are not typed by the kind of entity that they may name. It is possible that there are different sorts of objects that may be able to serve the same needs and therefore might legitimately be gathered under a single generic name. Consider for example name translation. In some cases this may be achieved by invocation of operations on a table. In this case, a subroutine call may be made in the user's process. In other situations, there may be a daemon running, and a name is translated by sending a message to the daemon requesting the translation. Using most naming facilities, it is not possible to create under a single name a group of entities of such varying types as tables with code to guard them and daemons of which to make requests. Thus we conclude that in addition to the five original requirements developed for a single application, the requirement of a single unified naming facility is also of primary importance.

6. Summary

This paper began by introducing the concepts of *autonomy* and *federation* as they are applied to computer systems. This led to a discussion of requirements on a naming facility that would provide for the user an environment more like what is used in interpersonal communication. From there we considered a particular scenario that presented the problems of communicating among humans using a federation of computers as the medium of communication. We wanted to consider the facilities needed in a computer in order to allow the human communication to proceed in approximately the manner it would without computers. To this end, we proposed a new kind of object, the *conversation* composed of *contexts*. We then considered some other examples of situations in which entities need to be named, and found that there were the same set of requirements and the the same mechanisms would suffice. From this we concluded that a single mechanism would serve us better than what has been provided in the past.

During this paper we have made several important observations. The first is the set of requirements for a naming facility:

- multiplicity of names
- locality of names
- manifestation of meaning
- flexibility of names
- useability of names
- unification of the naming facility

The second important set of observations is the set of problems identified. These are the "reply-to" problem, the "name-equality" problem, the "who-is" problem, and the "mobile-name" problem. The set of requirements and problems led us to proposing contexts and conversations as a solution. Other mechanisms might address these issues at least as well and need further exploration.

References

[1]

D. H. Crocker, J. J. Vittal, K. T. Pogran, D. A. Henderson.
Standard for the Format of ARPA Network Text Mesages.
ARPANET RFC 733, SRI International, November, 1977.
NIC #41952

[2]

D. Farber and J. J. Vittal.
Extendability Considerations in the Design of the Distributed Computer System.
In *Proceedings of the National Telecommunications Conference*, pages 15E1 - 15E6. IEEE A & E S Society, IEEE Communications Society, and the IEEE G & E Group, Atlanta, Georgia, November, 1973.

[3]

D. K. Gifford.
Weighted Voting for Replicated Data.
In *Proc. of the Seventh Symposium on Operating Systems*, pages 150-162. ACM/SIGOPS, Pacific Grove, Calif., December, 1979.

[4]

B. Lindsay.
Object Naming and Catalog Management for a Distributed Database Manager.
Research Report RJ2914, IBM Research Laboratory, August, 1980.

[5]

J. H. Saltzer, D. P. Reed, D. C. Clark.
Source Routing for Campus Wide Internet Transport.
In A. West and P. Janson, editors, *Local Networks for Computer Communications*, pages 1-23. IFIP, North Holland Publishing Co., New York, N. Y., 1981.
Also Proceedings of the IFIP Working Group 6.4 International Workshop on Local Networks, organized by IBM, Zurich, Switzerland, August. 27-29, 1980

[6]

Z. Su and J. Postel.
The Domain Naming Conventions for Internet User Applications.
ARPANET RFC 819, SRI International, August, 1982.

[7]

C. Sunshine.
Source Routing in Computer Networks.
Computer Communications Review 1(7):29-33, January, 1977.