```
09:22:37            OCA PAD AMENDMENT - PROJECT HEADER INFORMATION            11/20/95

                                                                               Active
Project #: C-36-X19            Cost share #:                          Rev #: 4
Center # : 10/24-6-R7941-0A0   Center shr #:                          OCA file #:
                                                                      Work type : RES
Contract#: NCR-9305115                  Mod #: BR DTD 951116          Document  : GRANT
Prime    #:                                                           Contract entity: GTRC

Subprojects ? : N                                                     CFDA:
Main project #:                                                       PE #:


Project unit:                 COMPUTING     Unit code: 02.010.300
Project director(s):
   AMMAR M H                  COMPUTING     (404)894-3292



Sponsor/division names: NATL SCIENCE FOUNDATION        / GENERAL
Sponsor/division codes: 107                            / 000


Award period:      930801     to     970131  (performance)    970430  (reports)

Sponsor amount            New this change                Total to date
     Contract value                0.00                      171,973.00
     Funded                        0.00                      171,973.00
Cost sharing amount                                                0.00

Does subcontracting plan apply ?: N

Title: MULTI-PROTOCOL ARCHITECTURE AS A PARADIGM FOR ACHIEVING INTER-OPERABILITY


                              PROJECT ADMINISTRATION DATA

OCA contact: Jacquelyn L. Bendall       894-4820

 Sponsor technical contact                  Sponsor issuing office

 DARLEEN L. FISHER                          ALFRED W. WILSON
 (202)357-9717                              (703)306-1212

 NATIONAL SCIENCE FOUNDATION                NATIONAL SCIENCE FOUNDATION
 4201 WILSON BOULEVARD                      4201 WILSON BOULEVARD
 ARLINGTON, VA 22230                        ARLINGTON, VA 22230


Security class (U,C,S,TS) : U              ONR resident rep. is ACO (Y/N): N
Defense priority rating   :   N/A          NSF supplemental sheet
Equipment title vests with:    Sponsor          GIT X

Administrative comments -
  ISSUED TO EXTEND PERIOD OF PERFORMANCE TO JANUARY 31, 1997 WITH THE NSF FINAL
  REPORT DUE APRIL 30, 1997 VIA OPAS FORM.
```

2-SR460

Georgia Institute of Technology
Office of Contract Administration
PROJECT CLOSEOUT - NOTICE

07-MAY-1997 12:53

4

2

Closeout Notice Date 07-MAY-1997

Project Number C-36-X19

Doch Id 45352

Center Number 10/24-6-R7941-0A0

Project Director AMMAR, MOSTAFA

Project Unit COMPUTING

Sponsor NATL SCIENCE FOUNDATION/GENERAL

Division Id 3393

Contract Number NCR-9305115

Contract Entity GTRC

Prime Contract Number

Title MULTI-PROTOCOL ARCHITECTURE AS A PARADIGM FOR ACHIEVING INTER-OPERABILITY

Effective Completion Date 31-JAN-1997 (Performance) 30-APR-1997 (Reports)

Closeout Action:	Y/N	Date Submitted
Final Invoice or Copy of Final Invoice	N	
Final Report of Inventions and/or Subcontracts	N	
Government Property Inventory and Related Certificate	N	
Classified Material Certificate	N	
Release and Assignment	N	
Other	N	

Comments
LETTER OF CREDIT APPLIES. 98A SATISFIES PATENT REPORT.

Distribution Required:	
Project Director/Principal Investigator	Y
Research Administrative Network	Y
Accounting	Y
Research Security Department	N
Reports Coordinator	Y
Research Property Team	Y
Supply Services Department	Y
Georgia Tech Research Corporation	Y
Project File	Y

C-36-X19

1

APPENDIX D

ANNUAL NSF GRANT PROGRESS REPORT

NSF Program:	NCR	**NSF Award Number:**	NCR-9305115
PI Name:	Mostafa Ammar Kenneth Calvert	**Period Covered By This Report:**	8-1-93/ 7-31-94
PI Institution:	Georgia Tech	**Date:**	April 27, 1994
PI Address:	College of Computing, Georgia Tech, Atlanta, GA 30332-0280		

[✓] **Check if Continued Funding is Requested**

Please include the following information:

1. Brief summary of progress to date and work to be performed during the succeeding period;
2. Statement of funds estimated to remain unobligated —if more than 20%— at the end of the period for which NSF currently is providing support (not required for participants in the Federal Demonstration Project);
3. Proposed budget for the ensuing year in the NSF format, only if the original award letter did not indicate specific incremental amounts or if adjustments to a planned increment exceeding the greater of 10% or $10,000 are being requested;
4. Current information about other research support of senior personnel, if changed from the previous submission;
5. Any other significant information pertinent to the type of project supported by NSF or as specified by the terms and conditions of the grant;
6. A statement describing any contribution of the project to the area of education and human-resource development, if changed from any previous submission; and
7. Updated information on animal care and use, Institutional Biohazard Committee and Human Subject Certification, if changed substantially from those originally proposed and approved.

I certify that to the best of my knowledge (1) the statements herein (excluding scientific hypotheses and scientific opinions) are true and complete, and (2) the text and graphics in this report as well as any accompanying publications or other documents, unless otherwise indicated, are the original work of the signatories or individuals working under their supervision. I understand that the willful provision of false information or concealing a material fact in this report or any other communication submitted to NSF is a criminal offense (U.S. Code, Title 18, Section 1001.)

P.I. Signature: [illegible]

NSF Form 1328 (1/94)

Annual NSF Grant Progress Report

Multiprotocol Architecture as a Paradigm for Achieving Interoperability

NSF Program:	NCR	**NSF Award Number:**	NCR-9305115
PI Names:	Mostafa H. Ammar Kenneth L. Calvert	**Report Period:**	93/08/01 - 94/07/31
PI Institution:	Georgia Tech	**Date:**	April 26, 1994
PI Address:	College of Computing, Georgia Tech, Atlanta, GA 30332-0280		

1 Progress to Date

This project has had three major thrusts this year:

- Solutions to the protocol determination problem;
- Protocol features that support and enhance Multiprotocol Interoperability;
- Understanding how protocol implementation frameworks affect the cost of gaining interoperability.

1.1 Protocol Determination

The *protocol determination problem* is the following: given peer applications that need to communicate, and communication subsystems that implement (possibly different, but not necessarily nonintersecting) sets of protocols, choose a particular protocol configuration (stack) that (i) implements the service required by the application, and (ii) is supported on both systems. While in most cases today the problem is trivial, the interesting case—where the host that initiates communication has a nontrivial choice—is becoming more common.

We have developed and implemented an approach to this problem that extends the solution to the similar problem of determining the network address of a particular host or server. The basic idea is to view that portion of a host's protocol graph above the network layer as a part of the network topology, and to use the directory service to maintain information about what paths through that protocol graph are supported by the host.

One of the philosophical tenets of this project is that the number of architectural details that must be universally supported should be minimized. From this viewpoint, this directory-based approach is attractive because it does *not* require universal support for a single directory service; the only aspect on which global agreement is required is the form of the representation(s) of the protocol graph in the directory. We have designed such a representation for use with the Internet Domain Name service; it has been implemented by graduate student Russ Clark. The results of this work are described in a paper to be presented in June at INFOCOM '94 [2], and in a paper submitted to the IEEE *Journal on Selected Areas of Communications* special issue on the Global Internet [3].

1.2 Multiprotocol-Friendly Features

We have identified a number of *protocol features* that are desirable from the point of view of supporting interoperability across stacks and architectures. We submitted an Internet Draft [4] describing features of this kind for the next generation Internet Protocol in response to the IETF Task Force's call for white papers. In this draft, we stressed the importance of multiprotocol support to lessen the perceived risk of migration to the next generation protocol, and thus speed its penetration. Among the characteristics highlighted in the internet draft are: support for variable address formats in general and embeddable addresses in particular; large address spaces for multiplexing, and control over the amount and nature of feedback reports received from the network.

1.3 Protocol Frameworks and Interoperability

The third thrust of our work considers determinants of the cost of *gaining interoperability* between systems that do not initially have it. In other words, given two systems supporting distinct sets of protocols, whose intersection is inadequate to provide some required service, what are the impediments to arriving at a state where both systems support an adequate set? And how can those impediments be removed? It is well known that this has as much or more to do with portability of protocol implementations across operating systems as with the protocols themselves. Current frameworks for protocol subsystems tend to support easy from-scratch implementations as opposed to porting existing protocol implementations (e.g., Streams), or emphasize high performance, or require that all communicating systems support the same architecture to take full advantage of its features (e.g. *x*-kernel, Morpheus). We are investigating ways to design such frameworks with an eye toward mixing and matching existing and new protocols. So far our efforts have focused on gaining experience with two existing frameworks, Streams and the *x*-kernel. Two of our graduate students, Rich Clayton and Bobby Krupczak, have implemented substantial portions of the Appletalk protocol stack, one in Streams and one in the *x*-kernel. A paper comparing the two frameworks based on this experience is now in preparation [5].

2 Plans for Next Year

Our plans for the next year include:

- *Protocol Determination* — investigate non-directory-service-based solutions to the protocol determination problem.
- *Protocol Frameworks* — design and begin implementation of a new protocol implementation framework designed with *both* interoperability *and* performance in mind.
- *Formal Specifications and Interoperability* — investigate the utility of formal specifications as a means of reducing the cost of gaining interoperability, along the lines

discussed in [6].

3 Other Research Support of PIs

3.1 M. H. Ammar

Current:

Principal Investigator (with M. Ahamad), "Using Replication to Build High Performance Distributed Systems," ARPA/CSTO, September 1993 – August 1995, approx. $200,000, 2 months/year.

Co-Principal Investigator (Melody Moore-PI, Navathe, Mark, McCracken, and Rugaber Co-PI.), "Transitioning to Open Systems Environments," Army Research Lab., August 1993 - July 1994, approx. $400,000, 2 months/year.

Pending:

Co-Prinicpal Investigator with K. Schwan, "High-Performance Application-Specific Protocol Architectures with Reconfigurable Functionality," submitted to NSF, December 1993, duration: 3 years, approx. $330,000, 1 month/year.

3.2 K. L. Calvert

Current:

Co-Investigator (Melody Moore-PI), "Transitioning to Open Systems Environments", Army Research Lab., August 1993 - July 1994, approx. $400,000, 1 month/year.

Pending:

Principal Investigator, NSF National Young Investigator Award.

4 Contribution to Education and Human Resources

We now have three full-time PhD students working on this project. One of them, Russell Clark, presented a disseration proposal, "A Multiprotocol Interoperability Architectural Framework for Achieving Interoperability" based on work performed under this project. The other two students, Richard Clayton and Bobby Krupczak, are just getting started in research programs.

References

[1] Russel J. Clark, Mostafa H. Ammar, and Kenneth L. Calvert. Multi-protocol architectures as a paradigm for achieving inter-operability. In *Proceedings IEEE INFOCOM '93, San Francisco*, March 1993.

[2] Russel J. Clark, Mostafa H. Ammar, and Kenneth L. Calvert. On the use of directory services to support multiprotocol interoperability. In *Proceedings IEEE INFOCOM '94, Toronto*, June 1994.

[3] Russell J. Clark, Mostafa H. Ammar, and Kenneth L. Calvert. The multiprotocol interoperability paradigm and its support using directory services. Submitted to *IEEE JSAC* special issue on the Global Internet, January 1994.

[4] Russell J. Clark, Mostafa H. Ammar, and Kenneth L. Calvert. Multiprotocol interoperability in IPng. Internet Draft draft-clark-ipng-multipro-interop-00.txt, January 1994.

[5] R. Clayton, R. Krupczak, K. Calvert, and M. Ammar. Implementation-based comparison of the x-kernel and Streams protocol subsystems. in preparation, 1994.

[6] Kenneth L. Calvert. Beyond layering: Modularity considerations for protocol architectures. In *Proceedings International Conference on Network Protocols, San Francisco*, October 1993.

Internet Engineering Task Force
INTERNET-DRAFT

Russell J Clark
Mostafa H Ammar
Kenneth L Calvert
Georgia Institute of Technology
January, 1994

Multiprotocol Interoperability In IPng
<draft-clark-ipng-multipro-interop-00.txt>

1 Status of this Memo

2 Executive Summary

The two most commonly cited issues motivating the introduction of IPng are address depletion and routing table growth in IPv4. Further motivation is the fact that the Internet is witnessing an increasing diversity in the protocols and services found in the network. When evaluating alternatives for IPng, we should consider how well each alternative addresses the problems arising from this diversity. In this document, we identify several features that affect a protocol's ability to operate in a multiprotocol environment and propose the incorporation of these features into IPng.

Our thesis, succinctly stated, is: The next generation Internet

Protocol should have features that support its use with a variety of protocol architectures.

3 Introduction

The Internet is not a single protocol network [4]. While TCP/IP remains the primary protocol suite, other protocols (e.g., IPX, AppleTalk, OSI) exist either natively or encapsulated as data within IP. As new protocols continue to be developed, we are likely to find that a significant portion of the traffic in future networks is not from single-protocol communications. It is important to recognize that multiprotocol networking is not just a transition issue. For instance, we will continue to see tunneling used to carry IPX traffic over the Internet between two Novell networks. Furthermore, the introduction of IPng is not going to result in a near term elimination of IPv4. Even when IPng becomes the primary protocol used in the Internet, there will still be IPv4 systems in use. We should consider such multiprotocol uses of the network as we design future protocols that can efficiently handle mixed protocol traffic.

We have identified several issues related to the way in which protocols operate in a multiprotocol environment. Many of these issues have traditionally been deemed ''less important'' by protocol designers since their goal was to optimize for the case where all systems supported the same protocol. With the increasing diversity of network protocols, this approach is no longer practical. By addressing the issues outlined in this paper, we can simplify the introduction of IPng to the Internet and reduce the risk for network managers faced with the prospect of supporting a new protocol. This will result in a faster, wider acceptance of IPng and increased interoperability between Internet hosts. In addition, by designing IPng to address these issues, we will make the introduction of future protocols (IPng2) even easier.

The outline for this document is as follows. In Section 4 we motivate the issues of multiprotocol networking with a discussion of an example system. In Section 5 we describe three main techniques for dealing with multiple protocols. This is followed in Section 6 by a description of the various protocol features that are important for implementing these three techniques. We conclude in Section 7 with a summary of the issues raised.

4 Multiprotocol Systems

Consider the multiprotocol architecture depicted in Figure 1. A system supporting this architecture provides a generic file-transfer service using either the Internet or OSI protocol stacks. The generic service presents the user with a consistent interface, regardless of the actual protocols used. The user can transfer files between this host and hosts supporting either of the single protocol stacks presented in Figures 2a and 2b. To carry out this file transfer, the user is not required to decide which protocols to use or to adjust between different application interfaces.

```
+-----------------------------------+
|       File Transfer Service       |
+-----------+-----------------------+
|           |         FTAM          |
|           +-----------------------+
|    FTP    |       ISO 8823        |
|           +-----------------------+
|           |       ISO 8327        |
|           +-----------+-----------+
|           |TP0/RFC1006|    TP4    |
+-----------+-----------+           |
|          TCP          |           |
+-----------+-----------+-----------+
|    IP     |         CLNP          |
+-----------+-----------------------+
```

Figure 1: Multiprotocol architecture providing file-transfer service

```
+-----------+    +-----------+    +-----------+    +-----------+
|    FTP    |    |   FTAM    |    |   FTAM    |    |    FTP    |
+-----------+    +-----------+    +-----------+    +-----------+
|    TCP    |    | ISO 8823  |    | ISO 8823  |    |    TCP    |
+-----------+    +-----------+    +-----------+    +-----------+
|    IP     |    | ISO 8327  |    | ISO 8327  |    |   CLNP    |
+-----------+    +-----------+    +-----------+    +-----------+
                 |    TP4    |    |TP0/RFC1006|
                 +-----------+    +-----------+
                 |   CLNP    |    |    TCP    |
                 +-----------+    +-----------+
                                  |    IP     |
                                  +-----------+

   a) TCP/IP        b) OSI          c) RFC 1006      d) TUBA
```

Figure 2: Protocol stacks providing file-transfer service.

Figure 2c depicts a mixed stack architecture that provides the upper layer OSI services using the Internet protocols. This is an example of a ''transition architecture'' for providing OSI applications without requiring a full OSI implementation. Figure 2d depicts a mixed stack architecture that provides the upper layer Internet applications using the OSI network protocol. In addition to communicating with the two previous simple protocol stacks, the

multiprotocol system of Figure 1 includes all the protocols necessary to communicate with these two new, mixed protocol stacks.

It is likely that many future network systems will be configured to support multiple protocols including IPng. As the IPng protocol is deployed, it is unreasonable to expect that users will be willing to give up any aspect of their current connectivity for the promise of a better future. In reality, most IPng installations will be made ''in addition to'' the current protocols. The resulting systems will resemble Figure 1 in that they will be able to communicate with systems supporting several different protocols.

Unfortunately, in most current examples, the architecture of Figure 1 is implemented as independent protocol stacks. This means that even though both TCP and CLNP exist on the system, there is no way to use TCP and CLNP in the same communication. The problem with current implementations of architectures like Figure 1 is that they are designed as co-existence architectures and are not integrated interoperability systems. We believe future systems should include mechanisms to overcome this traditional limitation. By integrating the components of multiple protocol stacks in a systematic way, we can interoperate with hosts supporting any of the individual stacks as well as those supporting various combinations of the stacks.

In order to effectively use multiple protocols, a system must identify which of the available protocols to use for a given communication task. We call this the Protocol Determination [2] task. In performing this task, a system determines the combination of protocols necessary to provide the needed service. For achieving interoperability, protocols are selected from the intersection of those supported on the systems that must communicate.

5 Multiprotocol Techniques

In this section we identify three main techniques to dealing with multiprotocol networks that are in use today and will continue to be used in the Internet. The first two techniques, tunneling and conversion, are categorized as intermediate-system techniques in that they are designed to achieve multiprotocol support without changing the end-systems. The third technique explicitly calls for the support of multiple protocols in end-systems. By describing these techniques here, we can motivate the need for the specific protocol features described in Section 6.

5.1 Encapsulation/Tunneling

Encapsulation or tunneling is commonly used when two networks that support a common protocol must be connected using a third intermediate network running a different protocol. Protocol packets from the two end networks are carried as data within the protocol of the intermediate network. This technique is only appropriate when both end-systems support the same protocol stack. It does not provide interoperability between these end systems and systems that only support the protocol stack in the intermediate network. Some examples of this technique are: a mechanism for providing the OSI transport services on top of the Internet protocols [13], encapsulating IEEE 802.2 frames in IPX network packets [5], tunneling IPX [10] and AppleTalk traffic over the Internet backbone. We expect IPng to be used for tunneling other network protocols over IPng and to be encapsulated.

5.2 Translation/Conversion

Despite their known limitations [8], translation or conversion gateways are another technique for handling multiple protocols [11, 12]. These gateways perform direct conversion of network traffic from one protocol to another. The most common examples of conversion gateways are the many electronic mail gateways now in use in the Internet. In certain cases it may also be feasible to perform conversion of lower layer protocols such as the network layer. This technique has been suggested as part of the transition plan for some of the current IPng proposals [3, 15].

5.3 Multiprotocol End-Systems

We expect that IPng will be introduced as an additional protocol in many network systems. This means that IPng should be able to coexist with other protocols on both end- and intermediate-systems. Specifically, IPng should be designed to support the Protocol Determination task described in Section 4.

One technique that we consider for solving the Protocol Determination problem is to employ a directory service in distributing system protocol configuration information. We have developed and implemented mechanism for using the Internet Domain Name System (DNS) [6, 7] to distribute this protocol information [2]. Using this mechanism, a multiprotocol host can determine the protocol configuration of a desired host when it retrieves the network address for that host. Then the multiprotocol host can match the configuration of the desired host to its own configuration and determine which protocols should be used to carry out the requested communication service.

Another alternative to determining protocol information about another host is Protocol Discovery. Using this approach, a host determines which protocols to use by trial-and-error with the protocols currently available. The initiating host monitors successive attempts to communicate and uses the information gained from that monitoring to build a knowledge base of the possible protocols of the remote system.

This knowledge is used to determine whether or not a communication link can be established and if it can, which protocol should be used.

An important aspect of the Protocol Discovery approach is that it requires an error and control feedback system similar to ICMP [9], but with additional functionality (See Section 6).

6 Protocol Features

In this section we identify features that affect a protocol's ability to support the multiprotocol techniques described in the previous section. These features indicate specific areas that should be considered when comparing proposed protocols. We present two different types of protocol features: those that should be included as part of the IPng protocol standard, and those that should be considered as part of the implementation and deployment requirements for IPng.

6.1 Protocol Standard Features

o Addressing

A significant problem in dealing with multiprotocol networks is that most of the popular network protocols use different addressing mechanisms. The problem is not just with different lengths but also with different semantics (e.g. hierarchical vs. flat addresses). In order to accommodate these multiple formats, IPng should have the flexibility to incorporate many address formats within its addressing mechanism.

A specific example might be for IPng to have the ability to include an IPv4 or IPX address as a subfield of the IPng address. This would reduce the complexity of performing address conversion by limiting the number of external mechanisms (e.g., lookup tables) needed to convert an address. This reduction in complexity would facilitate both tunneling and conversion. It would also simplify the task of using IPng with legacy applications which rely on a particular address format.

o Header Option Handling

In any widely used protocol, it is advantageous to define option mechanisms for including header information that is not required in all packets or is not yet defined. This is especially true in multiprotocol networks where there is wide variation in the requirements of protocol users. IPng should provide efficient, flexible support for future header options. This will better accommodate the different user needs and will facilitate conversion between IPng and other protocols with different standard features.

As part of the support for protocol options, IPng should include a mechanism for specifying how a system should handle unsupported options. If a network system adds an option header, it should be able to specify whether another system that does not support the option should drop the packet, drop the packet and return an error, forward it as is, or forward it without the option header. The ability to request the ``forward as is'' option is important when conversion is used. When two protocols have different features, a converter may introduce an option header that is not understood by an intermediate node but may be required for interpretation of the packet at the ultimate destination. On the other hand, consider the case where a source is using IPng with a critical option like encryption. In this situation the user would not want a conversion to be performed where the option was not understood by the converter. The ``drop the packet'' or ``drop and return error'' options would likely be used in this scenario.

o Multiplexing

The future Internet protocol should support the ability to distinguish between multiple users of the network. This includes the ability to handle traditional ``transport layer'' protocols like TCP and UDP, as well as other payload types such as encapsulated AppleTalk packets or future real-time protocols. This kind of protocol multiplexing can be supported with an explicit header field as in IPv4 or by reserving part of the address format as is done with OSI NSEL's.

In a multiprotocol network there will likely be a large number of different protocols running atop IPng. It should not be necessary to use a transport layer protocol for the sole purpose of providing multiplexing for the various network users. The cost of this additional multiplexing is prohibitive for future high-speed networks [14]. In order to avoid the need for an additional level of multiplexing, the IPng should either use a payload selector larger than the 8-bits used in IPv4 or provide an option for including additional payload type information within the header.

o Status/Control Feedback

With multiple protocols, the correct transmission of a packet might include encapsulation in another protocol and/or multiple conversions to different protocols before the packet finally reaches its destination. This means that there are many different places the transmission can fail and determining what went wrong will be a challenge.

In order to handle this situation, a critical protocol feature in multiprotocol networks is a powerful error reporting mechanism.

In addition to reporting traditional network level errors, such as those reported by ICMP [9], the IPng error mechanism should include feedback on tunneling and conversion failures. Also, since it is impossible to know exactly which part of a packet is an encapsulated header, it is important that the feedback mechanism include as much of the failed packet as possible in the returned error message.

In addition to providing new types of feedback, this mechanism should support variable resolution such that a transmitting system can request limited feedback or complete information about the communication process. This level of control would greatly facilitate the Protocol Discovery process described in Section 5.3. For example, a multiprotocol system could request maximal feedback when it sends packets to a destination it has not communicated with for some time. After the first few packets to this ``new'' destination, the system would revert back to limited feedback, freeing up the resources used by the network feedback mechanisms.

Finally, it is important that the information provided by the feedback mechanism be available outside the IPng implementation. In multiprotocol networks it is often the case that the solution to a communication problem requires an adjustment in one of the protocols outside the network layer. In order for this to happen, the other protocols must be able to access and interpret these feedback messages.

o MTU Discovery or Fragmentation

A form of multiprotocol support that has long been a part of networking is the use of diverse data link and physical layers. One aspect of this support that affects the network layer is the different Maximum Transmission Units (MTU) used by various media formats. For efficiency, many protocols will attempt to avoid fragmentation at intermediate nodes by using the largest packet size possible, without exceeding the minimum MTU along the route. To achieve this, a network protocol performs MTU discovery to find the smallest MTU on a path.

The choice of mechanism for dealing with differing MTUs is also important when doing conversion or tunneling with multiple protocols. When tunneling is performed by an intermediate node, the resulting packets may be too large to meet the MTU requirements. Similarly, if conversion at an intermediate node results in a larger protocol header, the new packets may also be too large. In both cases, it may be desirable to have the source host reduce the transmission size used in order to prevent the need for additional fragmentation. This information could be sent to the source host as part of the previously described feedback mechanism or as an additional MTU discovery message.

6.2 Implementation/Deployment Features

o Switching

We define switching in a protocol as the capability to simultaneously use more than one different underlying protocol [1]. In network layer protocols, this implies using different datalink layers. For example, it may be necessary to select between the 802.3 LLC and traditional Ethernet interfaces when connecting a host to an "ethernet" network. Additionally, in some systems IPng will not be used directly over a datalink layer but will be encapsulated within another network protocol before being transmitted. It is important that IPng be designed to support different underlying datalink services and that it provide mechanisms allowing IPng users to specify which of the available services should be used.

o Directory Service Requirements

While not specifically a part of the IPng protocol, it is clear that the future Internet will include a directory service for obtaining address information for IPng. In light of this, there are some features of the directory service that should be considered vis-a-vis their support for multiple protocols.

First, the directory service should be able to distribute address formats for several different protocol families, not just IPng and IPv4. This is necessary for the use of tunneling, conversion, and the support of multiprotocol systems. Second, the directory service should include support for distributing protocol configuration information in addition to addressing information for the network hosts. This feature will support the protocol determination task to be carried out by multiprotocol systems [2].

7 Conclusion

Future networks will incorporate multiple protocols to meet diverse user requirements. Because of this, we are likely to find that a significant portion of the traffic in the Internet will not be from single-protocol communications (e.g. TCPng/IPng). This will not just be true of near term, transitional networks but will remain as a reality for most of the Internet. As we pursue the selection of IPng, we should consider the special needs of multiprotocol networks. In particular, IPng should include mechanisms to handle mixed protocol traffic that includes tunneling, conversion, and multiprotocol end-systems.

8 Acknowledgments

The authors would like to acknowledge the support for this work by a grant from the National Science Foundation (NCR-9305115) and the TRANSOPEN project of the Army Research Lab (formerly AIRMICS) under contract number DAKF11-91-D-0004.

9 References

[1] R. J. Clark, M. H. Ammar, and K. L. Calvert. Multi-protocol architectures as a paradigm for achieving inter-operability. In Proceedings of IEEE INFOCOM, April 1993.

[2] R. J. Clark, K. L. Calvert, and M. H. Ammar. On the use of directory services to support multiprotocol interoperability. To appear in proceedings of IEEE INFOCOM, 1994. Technical Report GIT-CC-93/56, College of Computing, Georgia Institute of Technology, ATLANTA, GA 30332-0280, August 1993.

[3] R. Gilligan, E. Nordmark, and B. Hinden. IPAE: the SIPP interoperability and transition mechanism. Internet Draft, November 1993.

[4] B. Leiner and Y. Rekhter. The multiprotocol internet. RFC 1560, December 1993.

[5] L. McLaughlin. Standard for the transmission of 802.2 packets over IPX networks. RFC 1132, November 1989.

[6] P. Mockapetris. Domain names - concepts and facilities. RFC 1034, November 1987.

[7] P. Mockapetris. Domain names - implementation and specification. RFC 1035, November 1987.

[8] M. A. Padlipsky. Gateways, architectures, and heffalumps. RFC 875, September 1982.

[9] J. B. Postel. Internet control message protocol. RFC 792, September 1981.

[10] D. Provan. Tunneling IPX traffic through IP networks. RFC 1234, June 1991.

[11] M. T. Rose. The Open Book. Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

[12] M. T. Rose. The ISO Development Environment User's Manual - Version 7.0. Performance Systems International, July 1991.

College of Computing, Georgia Tech, Atlanta, GA 30332-0280
rjc@cc.gatech.edu

[13] M. T. Rose and D. E. Cass. ISO Transport Services on top of the TCP. RFC 1006, May 1987.

[14] D. L. Tennenhouse. Layered multiplexing considered harmful. In IFIP Workshop on Protocols for High-Speed Networks. Elsevier, May 1989.

[15] R. Ullmann. CATNIP: Common architecture technology for next-generation internet protocol. Internet Draft, October 1993.

10 Authors' Address

Russell J Clark - rjc@cc.gatech.edu
Mostafa H Ammar - ammar@cc.gatech.edu
Kenneth L Calvert - calvert@cc.gatech.edu
College of Computing Georgia Institute of Technology
Atlanta, GA 30332-0280

----<END OF DOCUMENT>----
--
Russ Clark

Infocom '94

On the Use of Directory Services to Support Multiprotocol Interoperability*

Russell J. Clark, Kenneth L. Calvert, Mostafa H. Ammar
College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280

Abstract

Multiprotocol systems are a vital tool for achieving interoperability in today's heterogeneous communication networks. An important aspect of these systems is the need to determine which of the multiple available protocols will be used to carry out a given communication task; an uninformed choice can result in failure to communicate when communication should be possible. In this paper we consider ways to make information about hosts' supported protocol configurations available through directory services. We discuss various representation approaches, and describe a working implementation of a multiprotocol application exemplifying our approach.

1 Introduction

The past decade has seen the development and deployment of many different protocol architectures, including TCP/IP, SNA, DECNET, and IPX. Radical growth in the number of interconnected systems has accompanied this proliferation of protocol suites. As a result of these two facts, there is an ever-increasing need to interconnect systems that do not currently use the same communication protocols. In a previous paper [3] we discussed support for multiple protocol suites as a method of achieving interoperability in current and future networks. Such an approach is based upon the observation that universal support for any single architecture or even protocol is unlikely. The basic idea is to embrace heterogeneity by developing frameworks for dealing with it. This paper presents one element of such a framework, namely a method enabling hosts to obtain information about the protocols supported by other hosts.

Multiprotocol networking has sparked research in several areas. Cypser [5] describes three main locations for protocol switching in systems supporting multiple protocols. Ogle et al. [10] are developing a TCP/IP and SNA system that performs protocol selection below the socket level interface. Janson et al. [6] consider options for interoperability between OSI and SNA networks, and analyze the addressing issues arising when these protocols are combined in a single network. Instead of an architecture-specific solution, however, we seek an approach general enough for problems involving future architectures as well as today's. Among others considering the general interoperability problem, Tschudin has described a "generic protocol" [15] allowing communicating systems to exchange descriptions of arbitrary protocols before using them to communicate. Similar in philosophy is the "meta-protocol" concept proposed by Meandzija [7]. In contrast, one of the explicit goals of our work is to minimize the number of things (including protocols) that must be universally agreed upon or supported.

This paper considers the use of *directory services* to provide information about the various protocols that hosts support. Such information is useful in a multiprotocol context because there may be more than one protocol configuration supporting a given communication service, and choosing the wrong configuration can lead to a failure to communicate. Using directory services to provide this information at runtime is a natural extension of the primary use of these services today, i.e. mapping names to address information. We describe a generic approach to providing protocol information, along with a working implementation based on a widely-used directory service, the Internet Domain Name System [8,9].

The paper proceeds as follows. In the next section, we describe the problem in more detail, along with the basic idea of the solution. In Section 3, we consider possible methods of encoding multiprotocol information in a directory database. General features of a directory service supporting this function are discussed in Section 4. In Section 5 we consider ways to implement these features using the existing Internet Domain Name System and describe a working multiprotocol application, incorporating an implementation of the proposed features. Finally, Section 6 contains some concluding remarks.

*This research is supported by a grant from the National Science Foundation (NCR-9305115) and the TRANSOPEN project of the Army Research Lab (formerly AIRMICS) under contract number DAKF11-91-D-0004.

2 The Protocol Environment

We consider a communication subsystem that provides services to users via an abstraction that hides details of the protocols from the user and serves as an endpoint for one communication instance. We shall refer to this abstraction as a *session*. The session contains information that determines which protocols process outgoing messages sent by a user, and that enables incoming messages from the network to be routed to the user. In addition, it may contain (pointers to) local state information required by the protocols.

A *protocol entity* (PE) is an abstract representation of a part of the communication subsystem that implements a particular protocol. A convenient description of a protocol entity is: "a component that adds a header to outgoing messages, and thus affects interoperability with other systems". Each session has associated with it a sequence of protocol entities, namely the protocol "stack" implementing the communication. The sequence of PEs associated with a session is here called a *path*.

The general scenario by which a user obtains network services through a session object involves the following steps:

1. Service Determination. Determine the *type of service* needed by the user. Note that "service" here may include such aspects as the format of the address by which the destination host is identified.

2. Path Determination. Determine the combination of protocols required to provide the needed service.

3. Path Configuration. Create a session object with the required protocols, determined in Step 2, and pass the session to the user.

4. Communication. The user performs the appropriate operations (open connections, send, receive, etc.) via the session.

Note that service determination may be performed by the user, or by some combination of user and communication subsystem. For example, the user may know the name of a desired destination, but not its address (or even its address format). To obtain the required information, either the user or the communication subsystem may make use of a *directory service*, which maps names to network addresses. Such a directory service may be a simple file local to the host, or a separate remote service, accessed via the network itself.

The path determination step is typically performed by the communication subsystem; however, it could be handled by the user *if* the subsystem allows the user to request a session object with any arbitrary protocol path.

The fact that different services require different combinations of protocols implies that the binding between (at least some) layers is delayed until the time of the user request. That is, some *switching* mechanism is required to enable a layer N protocol entity to be configured to use different layer $N-1$ protocols for different sessions. Within a single protocol suite, typically only one path corresponds to any given combination of service and network address; thus the need for switching is minimal, and the mapping from services to protocols *may* be fixed in advance by the communication subsystem. This is not necessarily true in multiprotocol environments.

2.1 Multiprotocol Complications

In general it does not make sense to combine protocols in arbitrary ways. There are at least two possible reasons why a PE a, at layer N, would be precluded from making use of another PE b at layer $N-1$.

One reason is architectural: if b does not provide the service required by a for correctness, then it simply makes no sense for a to be configured to run on top of b. For example, it is not sensible to run the OSI Class 0 Transport Protocol on top of CLNP, because the former requires a reliable, connection-oriented network service, while the latter provides a datagram service.

The other possible reason is that a particular *implementation* may not support the configuration of a atop b. In earlier versions of Berkeley UNIX, for example, the use of IP as a network protocol was hardwired into the TCP implementation; such implementations do not support the use of CLNP by TCP, even though conventions for such use have been defined [2]. In some cases, this constraint can be overcome by the addition of a switching capability to the implementation, in the form of a "pseudo PE" that provides a level of indirection to the actual lower-level PEs, and hides minor differences between their interfaces.

For a given set of PEs, the possible configurations that might be supported in a host can be represented by a directed graph with PEs as nodes (See Figure 1), in which the presence of an edge from a to b means that a can be configured to run on top of b. In this case we say *a uses b*.

Note. We are assuming here that selection of a lower-level PE is constrained *only* by the next higher-level PE. For example, if the paths $a \longrightarrow x \longrightarrow c$ and $b \longrightarrow x \longrightarrow d$ are both possible, then the path $a \longrightarrow x \longrightarrow d$ is also possible. One consequence of this is that the service provided to the user by a protocol path is fully determined by the topmost PE of the path. This assumption appears to be reasonable for existing protocols.

It was noted above that a single protocol suite generally has a unique path for a given service–address combination. This ensures that two hosts supporting the same suite can inter-operate and offer the service to their users, provided they both support the required protocols; if one host does not implement some required protocol, the service cannot be provided in any case. In an environment where protocol suites are mixed in some hosts, the situation is more complicated. The protocol graphs may contain multiple paths for the same service–address combination, and thus the Path Determination step involves selection of one of these paths. Moreover, installation graphs may differ from host to host; selection of the *wrong* path can prevent interoperability even in the case where the hosts support some other common path.

2.2 Path Determination Approaches

We have identified two possible approaches to the problem of Path Determination in a multiprotocol environment. The *directory-based* approach makes use of a database of information about the protocol graph supported by hosts on the network. The *path discovery* approach tries to establish communication using all of the possible paths, and monitors the results. This paper focuses on the directory-based approach.

Network directory services such as the Internet Domain Name System (DNS) or the OSI directory service (X.500) provide a distributed database of information about hosts, their addresses, and the applications they support. In current architectures this database is typically consulted to map host and service names to their respective network and application addresses during path determination. Adding information about a host's protocol paths to this database is thus a rather natural way to support path determination in a multiprotocol environment.

An important feature of the directory-based approach is that it does not require all hosts to make use of the directory service. For example, a host supporting only a single protocol suite need not refer to the directory's protocol path information at all, because its path determination problem is simple. To aid other multiprotocol hosts in establishing communication with the single-protocol host, information describing the protocol(s) it supports should be stored in the directory service, but no modification of its communication subsystem—or the way it uses the directory service, if any—is required. Note also that universal agreement on a single directory service is not required: it is only necessary that each multiprotocol host have access to a distributed database that contains *some* encoding of the protocol path information for the hosts with which it communicates. In the next section we consider some possibilities for such an encoding.

3 Encoding Protocol Path Information

We would like to encode information about a host's supported protocol graph in a form that enables another host to determine — at a minimum — whether a common path exists. Information about a host's supported protocol graph can be stored in a directory service in any of several forms; each of which requires that globally-understood identifiers be assigned to some parts of the graph structure. There are at least three levels at which such identifiers might be assigned: Protocol Entity, Protocol Path, Protocol Graph.

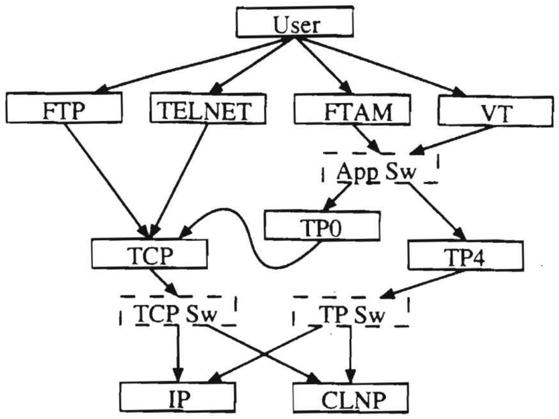

Figure 1: A multiprotocol graph.

To compare these approaches, we consider a host with the protocol graph of Figure 1. This host includes four different application PEs, three different transport PEs, and two network PEs[1]. The Figure also depicts the addition of a protocol switching function in three places: the Application Switch provides switching between the two OSI transport protocols, the TCP Switch provides TCP with the capability to select between IP and CLNP as a network service, and the TP Switch provides a similar function for TP4. These "pseudo-PEs" do not themselves implement protocols and do not add or modify message headers; they are included in the figure to emphasize that the protocols involved are designed to use a particular lower-level protocol, and do not support the switching function.

This graph supports ten different protocol paths: two for each TCP application and three for each OSI application. Let us consider the three protocol representation options as they would represent this particular example.

Protocol Entity: The first option involves assigning identifiers to the PEs themselves. The protocol graph structure can then be represented in any of several ways, e.g. by giving for each PE, a list of the PEs to which it has edges (its uses-list). The graph of Figure 1 would thus have an entry for "TCP" with a uses-list containing "IP" and "CLNP". The global identifiers

[1]A more precise depiction of this graph would include OSI Presentation and Session layers between the FTAM and VT applications and the Application Switch. For simplicity in this discussion, we omit these here.

are "TCP", "IP", etc. This approach can easily handle novel paths and configuration changes, and allows for partial matches when a complete matching path cannot be found. Partial matches might be useful in establishing a minimal level of communication with an unknown system.

Another way to assign identifiers at the PE level is to encode the PEs uses-list (i.e. its outgoing edges) in the identifier. Thus an implementation of TCP that can use both IP and CLNP would have a different identifier than one that can only use IP. The graph could then be represented as a simple list of PEs. This would generally yield a smaller representation, at the expense of a larger number of agreed-upon identifiers.

Protocol Path: The second representation option is to assign a standard identifier to every protocol path, and store a list of the paths supported. This approach simplifies path determination by providing a representation at the exact level where a match is sought. However, prior agreement on an identifier for every possible supported path is required; this may be a problem when new protocols and paths are introduced. This approach also provides a significant degree of redundant information because an application that uses multiple protocols will have a separate entry for each set of protocols it can use. For example, both the FTAM and VT applications in Figure 1 would be incorporated into three path entries.

Note that if the set of paths cannot be characterized by the uses-list relations of the PEs involved, i.e. the assumption of Section 2.1 is not valid, then this option may be preferable.

Protocol Graph: In this option, the protocol graph (or set of paths) itself is assigned a standard identifier. A significant problem with this representation approach is that it requires every existing protocol configuration to be known *a priori* so that a standard identifier can be assigned to it. This is infeasible given the diversity in the installed configurations of even single protocol suite systems.

From the above discussion, the protocol entity and path options appear to have the most potential. In order to more carefully compare the feasibility of the two approaches we examine the number of data items required to represent various protocol graphs. For the graph in Figure 1, assuming that a node's uses-list is encoded in its identifier, the number of stored items for these approaches is: nine PEs vs. ten protocol paths. When a new application is added using TCP, the PE approach requires the addition of a single PE entry. The path approach however, requires two additional path entries. If a new network layer PE were added, the PE approach would require the addition of one PE plus the addition or update of the transport PE that used this network PE. The path approach would require a new path entry for each application using this new protocol.

In general, the PE approach is more efficient whenever a host includes applications that can operate over multiple different protocols. Since multiprotocol systems are becoming more common, and indeed this is precisely the kind of system we are interested in supporting, the PE representation approach appears to be the most appropriate. As systems add more applications, the benefits of the PE approach are further realized.

4 An Ideal Directory Service

In this section we present the design requirements for a directory service that most effectively supports the protocol determination task. Our objective is to describe the necessary directory service features in a context which is free from the constraints of any current directory service products. Later we discuss how most of these features can be provided in the Internet Domain Name Service.

A problem in current directory service usage is the assumption that the availability of a particular network address for a host implies that the host supports a network protocol which utilizes that address. This assumption causes problems, for example, when translating gateways are used to provide transparent communication between two distinct protocols. In this scenario, the originating host must obtain an address for the destination that is compatible with the originating host's network protocol. For example, a host X which only supports IP cannot use a NSAP address to refer to another host Y even if X can communicate with Y through an IP/CLNP gateway. Host X will need an IP address to identify Y. An ideal multiprotocol directory service should, therefore, maintain network address information independent of protocol graph information. While it is true that before using a given network layer protocol it is necessary to obtain a network address for that protocol, the existence of a certain type of address for a system does not necessarily imply that the system directly supports any protocols which use that address.

Graph information is represented in the directory service as a collection of PEs and their uses-lists. The name of the PE is stored as a single string entry. The uses-list is stored as a string describing the set of PEs this PE can use. Conjunction and disjunction are indicated by the characters "&" and "|" respectively. Conjunction in a uses-list indicates that a PE requires the services of *both* underlying PEs to operate; e.g. OSI Presentation may require several Session Functional Units. Disjunction indicates that a PE can operate on any of the underlying PEs; e.g. the Transport Switch can select either IP or CLNP.

Table 1a presents the information desired in a directory service entry for the multiprotocol graph shown in Figure 1; Table 1b presents the graph of a single protocol host. PEs with an empty uses-list are known as *base* PEs. These indicate that no lower layer matching information is available for protocol paths that include this PE. In general, the network layer protocols will serve as base PEs.

PE Name	Uses-list
FTAM	TP0 | TP4
FTP	TCP
IP	
CLNP	
TP0	TCP
TP4	IP | CLNP
TCP	IP | CLNP
TELNET	TCP
VT	TP0 | TP4

a) Multiprotocol

PE Name	Uses-list
FTP	TCP
IP	
TCP	IP
TELNET	TCP

b) Single Protocol

Table 1: Protocol Graph Entries.

The two main functions of a directory service for multiprotocol systems are:

LookupHost(input: Hostname, output: AddressInfo, GraphInfo) This function retrieves the addressing and protocol graph information for the specified host from the directory service. The addressing information is returned as a collection of network addresses of various types. The graph information is returned as a collection of PEs with their uses-lists. The initiating host will invoke this routine once for the remote host and again to obtain its own local graph information.

MatchPath(input: GraphInfo, LocalGraphInfo, output: Path) This routine compares the two graphs and returns one or more common paths. The overall goal is to find a protocol path that is common to both graphs and will provide communication between the user application and a base PE. The exact return value and algorithm used is dependent upon the ultimate goal of the multiprotocol system. Achieving each of the goals we consider is equivalent in complexity to solving the subgraph-isomorphism problem. While the only known solutions to this problem are intractable for large graphs, the limited size of the protocol graphs coupled with the focused goals outlined below makes it feasible to solve this problem as part of communication establishment. The three possible path matching goals are:

• *Succeed or Fail:*
If the user is only interested in obtaining communication or finding out if communication is possible then a function that simply finds and returns the first successful match would suffice. This algorithm should start by matching a single PE and then try to build a single matching path.

• *All Matches:*
If a user wishes to be able to choose from multiple possible paths then it is necessary for the function to find all matches between the two graphs and return them. This function would be useful when there are several protocols supported by both hosts but one may be more appropriate for the given application. It is also possible that one or more of the valid paths may be temporarily unavailable due to a network failure. In this case the multiple paths would allow the user (or application) to try several different paths until one succeeds.

• *Partial Matches:*
In some cases there may not be a complete match found from the application all the way down to the base PE. In this case it may be appropriate to return partial match information about the PEs that did match. This would allow the system either to obtain a degraded level of communication or provide meaningful diagnostics to indicate exactly which components of the protocol architecture are missing. Partial matches might also be used as an aid in determining which gateway or translating bridge services might be useful in obtaining the desired communication. The algorithm for finding partial matches should be able to start anywhere in the protocol graph and find all PEs that match between the two graphs.

Each of the preceding path matching goals focus on finding paths that allow hosts to communicate. These goals could be further qualified to find paths that provide a particular service. This limits the matching algorithm to a specific PE or set of PEs with which to start the search and for which a path is considered valid.

5 A DNS-Compatible Implementation

The Internet Domain Name Service (DNS) is a popular example of the type of directory service that could provide protocol graph information. In this section we present an approach to using DNS to provide this extended service. In Section 5.3 we describe our implementation of this approach.

Our primary objective in this design is to develop a mechanism for delivering multiprotocol information that provides as many of the features identified in Section 4 as possible while minimizing the impact on current directory service implementations. Our approach requires that additional DNS support be provided only in multiprotocol systems that will take advantage of the new DNS features. The changes we propose have no impact on systems that currently use the DNS directory services. An alternative approach to using DNS would be to extend an X.500 implementation such as QUIPU, which is available with the ISO Development Environment (ISODE) [12]. While this approach would give us more flexibility to define new host information records, the ubiquity of DNS in the current Internet makes it more suitable for providing a system that could be deployed today.

5.1 An Overview of DNS

The DNS, described in [8] and [9], provides a hierarchically distributed database of network host information. It is used primarily to provide hostname to network address resolution. The two main components of the DNS are the *domain server* and the *resolver*. The domain server provides name service within a DNS domain. A domain corresponds to an administrative group such as a company or university. The resolver generally runs on the client host and provides the lookup service by successively querying domain servers. The actual data is stored on the server hosts in text files known as *master files*. The basic unit of information stored in the DNS is a *resource record* (RR). Each RR includes, among other things, a NAME field representing the node to which this entry pertains, a TYPE field representing the type of information stored, and an RDATA field representing the actual data for this entry.

Some important types of RRs are: `A` — the host address, `MX` — mail exchange information, `WKS` — the supported well known services, and `TXT` — a free-format text field. The `WKS` record format has a 32-bit address entry indicating the IP address, an 8-bit entry indicating a protocol, and a variable length bitmap indicating which services use that protocol. The protocol field contains the identifier of a protocol that uses IP such as TCP or UDP. The bitmap indicates which of the well known services are supported on the host: if a service is supported then the appropriate bit is set. These well known service numbers are used as port identifiers in the TCP and UDP protocols. For example, if FTP is supported then bit 21 is set since FTP uses port 21 of TCP. The protocol and well known service numbers are defined in the Internet Assigned Numbers document [11].

The standard interpretation of the protocol field in the `WKS` record is that it represents a transport layer protocol such as TCP. The service bitmap represents direct users of the transport layer. This interpretation is consistent with the TCP/IP network model where application services sit directly on top of the transport protocol and the well known service number is used as the port identifier in the transport protocol. The organization of this record implies that any service listed will use the transport protocol listed for this record; an application using TCP should be listed in a separate resource record from an application using UDP.

5.2 A Multiprotocol Usage of DNS

While the DNS was developed primarily for the TCP/IP environment, it has evolved to accommodate heterogeneous networks. For diversity at the network layer, a number of address formats have been defined. These address formats include an X.25 format, ISDN format, and an OSI style NSAP format. These are stored as RRs of TYPE `X25`, `ISDN`, and `NSAP` respectively. The RR type `A` is used for 32-bit IP addresses only.

An interesting aspect of the original design of DNS is the inclusion of a CLASS field in each resource record. This attribute is reserved for specifying information about the "supported protocol family" of a host [8]. The most natural extension of DNS to the multiprotocol environment is to use the CLASS field to designate which protocol architectures are supported. For instance, a class could be defined to indicate use of the OSI protocols. Unfortunately, this field has become largely meaningless in the current usage as only one value, "CLASS=IN" for Internet, has been widely used. Instead of designating different classes, each of the previously mentioned address type records has been created within the Internet class.

As we mentioned earlier, we are interested in developing a multiprotocol DNS that is compatible with most current DNS implementations. Our experience with current name server implementations, such as the BSD *named* program, is that they are largely hard coded for use with RR entries of class Internet. This means that the addition of a new CLASS value would require that current servers be modified to support the new classes and their associated types. We have not pursued this approach since this change would conflict with our goal of not requiring the replacement of current systems.

We have identified several possible approaches to using the current DNS architecture for distributing multiprotocol host information. All of these approaches use the alternative described in Section 4 where identifiers are assigned to PEs. They all use the currently defined IN class resource records. In this paper we present an approach that uses the `TXT` RRs. This approach provides the most flexibility in encoding and presents the least danger of conflicting with current implementation and usage. We describe some alternate approaches that use the `WKS` RR in the extended version of this paper [4].

5.3 A TXT Resource Record Approach

Currently, there is no widely used format for a `TXT` entry[2]. We propose that the `TXT` field be used to store a description of the PEs available on a host as well as the uses-lists of those PEs. The general format of the PE entries is "<PE>/<uses-list>". We use the leading string "PEInfo" to distinguish these protocol

[2]Rosenbaum has proposed a new mechanism for using TXT fields for arbitrary string attributes [13]. At the time of this writing, this was an experimental Internet standard.

descriptions from other TXT fields in use. A grammar for parsing these TXT entries is presented in [4].

Figure 2 shows a possible DNS entry for a host with the protocol graph given in Figure 1. This entry depicts the master file format used by the DNS server to store the domain information. The first two lines contain A and NSAP address RRs. The remaining lines contain TXT entries describing the host's supported protocols. Multiple PE entries in one TXT record are separated by a ";". If no uses-list is present, the entry is assumed to be a base PE. The entries are grouped by the protocol layer described. This organization is strictly for convenience when maintaining the file.

```
;Name Class RR-Type RR-Data
;-------------------------------------------------
mphost  IN  A       127.1.1.1
mphost  IN  NSAP    49.5100bd5a00
mphost  IN  TXT     "PEInfo:IP;CLNP"
mphost  IN  TXT     "PEInfo:TCP/IP|CLNP"
mphost  IN  TXT     "PEInfo:TP4/IP|CLNP;TP0/TCP"
mphost  IN  TXT     "PEInfo:FTP/TCP;TELNET/TCP"
mphost  IN  TXT     "PEInfo:FTAM/TP0|TP4;VT/TP0|TP4"
```

Figure 2: A multiprotocol DNS entry.

As a proof of concept, we will now describe a working multiprotocol FTAM implementation capable of carrying out a file transfer with hosts supporting one of several different protocol architectures. The three protocol paths supported by this implementation are shown in Figure 3. The three paths are FTAM using: TP4 over CLNP, TP4 over IP, and TP0 using RFC-1006 over TCP/IP. These paths are also present in the host described in the DNS entry of Figure 2. The basic steps carried out by the multiprotocol communication subsystem when attempting to provide the FTAM service are:

1. The user initiates a file transfer with the remote host by invoking FTAM with the remote host's name.

2. The communication subsystem obtains address and graph information by calling the DNS using the *LookupHost()* routine for both the remote and local hosts.

3. The communication subsystem calls the *MatchPath()* routine to obtain a subset of the two installation graphs that includes the FTAM service.

4. The communication subsystem configures the local host's installation graph to use the selected path for this session[3].

5. Control is passed back to the user application (FTAM) which continues normally by establishing a connection using the routines configured by the communication subsystem.

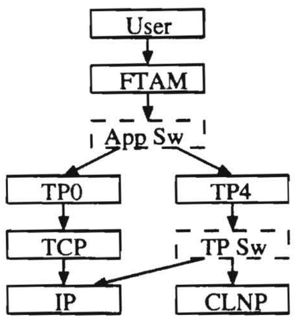

Figure 3: Multiprotocol FTAM Architecture.

Suppose that a user of this multiprotocol host wishes to perform a file transfer with the system "tcphost" described in Figure 4. This system only supports the single protocol path FTAM over TCP/IP using the RFC-1006 implementation of TP0. In this case, the only match found for performing file transfer is the "FTAM" entry. After matching the application, the next PE to match is "TP0", the only member of the "FTAM" uses-list for the host "tcphost". Next, "TCP" is matched and then "IP". Finally, once IP is selected, the A address entry is included for carrying out the communication.

```
;Name   Class RR-Type RR-Data
;--------------------------------------------
tcphost  IN  A       127.1.1.2
tcphost  IN  TXT     "PEInfo:IP;TCP/IP;TP0/TCP"
tcphost  IN  TXT     "PEInfo:FTAM/TP0"
```

Figure 4: A single protocol DNS entry.

To implement this system we developed a set of extensions to the BSD DNS resolver library and the ISODE 8.0 FTAM implementation. To the resolver we added the *LookupHost()* and *MatchPath()* functions discussed in Section 4. These extensions implement the path selection portion of the communication subsystem. The *LookupHost()* function retrieves the various address RRs and retrieves and parses the TXT fields to build the protocol graph information. *MatchPath()* compares the two protocol graphs and returns the first path found that supports the required service, which is FTAM in this case.

Once the path is selected, the system performs path configuration by creating a session for the user which instantiates this path. In our implementation, this task is performed by the Transport Switch mechanism provided by the ISODE [12]. The Transport Switch provides a mechanism to select among protocol stacks

[3]Here we use our definition of session from Section 2 rather than the OSI Session.

based on address information. For each transport connection, it associates a separate structure of function pointers to transport layer functions. In turn, these particular transport functions invoke specific network layer functions. To the ISODE switching architecture we added the ability to use the DNS routines to select which protocols should be used *without reliance on the address type*. In addition, we have added the ability to select TP4 over IP as one of the available paths. The use of this path is made possible by utilizing the protocol graph information from the directory service to select between two different paths, both of which use the IP network layer.

6 Concluding Remarks

In order to effectively use multiprotocol systems for communication in heterogeneous networks we must develop mechanisms for efficiently combining protocol architectures and managing their use. Managing the use of these systems involves both determining which protocols should be used and then specifying that usage to the systems. This paper describes our work in using a directory service to aid in the determination task. We discussed techniques for extending current single protocol architectures to operate in a multiprotocol environment. We described a practical extension of the current DNS for multiprotocol systems that involves no modification to the currently deployed DNS server software. We also presented a successful implementation of a multiprotocol application capable of using these extensions.

It should be noted that the examples used in this paper deal with protocols and implementations that exist today. However, we expect that new architectures will be developed, and that the problems considered here will become even more important as the next generation of architectures are deployed. Current proposals for the next generation Internet protocols all involve some sort of transition strategy where both current and future protocols will need to co-exist [14]. Our proposed architecture is an appropriate solution to this multiprotocol co-existence problem.

The question of accuracy of the DNS entries has been raised in the Internet community. Indeed the Internet host requirements document [1] specifically warns that a host should not rely on the **WKS** entries to provide accurate information regarding the services available from another host. These concerns over accuracy lead to the likelihood that the directory service itself may not be enough to provide up-to-date information about a network host. Our future work will look at ways to combine the directory service with a more dynamic discovery system that determines the supported protocols from the host itself when the directory service information is incomplete.

References

[1] R. Braden. Requirements for internet hosts - application and support. *RFC 1123*, October 1989.

[2] R. W. Callon. TCP and UDP with bigger addresses (TUBA), a simple proposal for internet addressing and routing. *RFC 1347*, June 1992.

[3] R. J. Clark, M. H. Ammar, and K. L. Calvert. Multi-protocol architectures as a paradigm for achieving inter-operability. In *Proceedings of IEEE INFOCOM*, April 1993.

[4] R. J. Clark, M. H. Ammar, and K. L. Calvert. On the use of directory services to support multiprotocol interoperability. Technical Report GIT-CC-93/56, College of Computing, Georgia Institute of Technology, ATLANTA, GA 30332-0280, September 1993.

[5] R. J. Cypser. Evolution of an open communications architecture. *IBM Systems Journal*, 31(2):161–188, 1992.

[6] P. Janson, R. Molva, and S. Zatti. Architectural directions for opening IBM networks: The case of OSI. *IBM Systems Journal*, 31(2):313–335, 1992.

[7] B. Meandzija. Integration through meta-communication. In *Proceedings of IEEE INFOCOM*, pages 702–709, June 1990.

[8] P. Mockapetris. Domain names - concepts and facilities. *RFC 1034*, November 1987.

[9] P. Mockapetris. Domain names - implementation and specification. *RFC 1035*, November 1987.

[10] D. M. Ogle, K. M. Tracey, R. A. Floyd, and G. Bollella. Dynamically selecting protocols for socket applications. *IEEE Network*, 7(3):48–57, May 1993.

[11] J. Reynolds and J. Postel. Assigned numbers. *RFC 1340*, July 1992.

[12] M. T. Rose. *The ISO Development Environment User's Manual - Version 7.0.* Performance Systems International, July 1991.

[13] R. Rosenbaum. Using the domain name system to store arbitrary string attributes. *RFC1464*, May 1993.

[14] IEEE Communications Society. Special issue: The future of the Internet Protocol. *IEEE Network*, 7(3), May 1993.

[15] C. Tschudin. Flexible protocol stacks. In *Computer Communication Review*, pages 197–205. ACM Press, September 1991.

C-36-X19 #2

OMB Number 345-0058

NATIONAL SCIENCE FOUNDATION
4201 Wilson Blvd.,
Arlington, VA 22230

BULK RATE
POSTAGE & FEES PAID
National Science Foundation
Permit No. G-69

PI/PD Name and Address

Mostafa H. Ammar
College of Computing
[redacted]
801 Atlantic Drive
Atlanta GA 30332-0280

NATIONAL SCIENCE FOUNDATION FINAL PROJECT REPORT

PART I - PROJECT IDENTIFICATION INFORMATION

1. Program Official/Org.	Tatsuya Suda - NCR
2. Program Name	NETWORKING RESEARCH
3. Award Dates (MM/YY)	From: 08/93 To: 01/97
4. Institution and Address	[redacted] Administration Building Atlanta GA 30332
5. Award Number	9305115
6. Project Title	Multi-Protocol Architecture as a Paradigm for Achieving Inter-Operability

You are encouraged to submit your Final Project Report electronically through the NSF FastLane home page (www.fastlane.nsf.gov).

This Packet Contains
NSF Form 98A
And 1 Return Envelope

NSF Grant Conditions (Article 17, GC-1, and Article 9, FDP-11) require submission of a Final Project Report (NSF Form 98A) to the NSF program officer no later than 90 days alter the expiration of the award. Final Project Reports for expired awards must be received before new awards can be made (NSF Grants Policy Manual Section 677).

elow, or on a separate page attached to this form, provide a summary of the completed projects and technical information. Be ure to include your name and award number on each separate page. See below for more instructions.

PART II - SUMMARY OF COMPLETED PROJECT (for public use)

The summary (about 200 words) must be self-contained and intelligible to a scientifically literate reader. Without restating the roject title, it should begin with a topic sentence stating the pro~~ject's major thesis. The summary~~ should include, if pertinent o the project being described, the following items:

The primary objectives and scope of the project
The techniques or approaches used only to the degree necessary for comprehension
The findings and implications stated as concisely and informatively as possible.

PART III - TECHNICAL INFORMATION (for program management use)

List references to publications resulting from this award and briefly describe primary data, samples, physical collections, inventions, software, etc. created or gathered in the course of the research and, if appropriate, how they are being made available o the research community. Provide the NSF Invention Disclosure number for any invention.

[redacted]

I certify to the best of my knowledge (1) the statements herein (excluding scientific hypotheses and scientific opinion) are true and complete, and (2) the text and graphics in this report as well as any accompanying publications or other documents, unless otherwise indicated, are the original work of the signatories or of individuals working under their supervision. I understand that willfully making a false statement or concealing a material fact in this report or any other communication submitted to NSF is a criminal offense (U.S. Code, Title 18, Section 1001).

[illegible]	5/1/97
Principal Investigator/Project Director Signature	**Date**

IMPORTANT:
MAILING INSTRUCTIONS
Return this *entire* packet plus all attachments in the envelope attached to the back of this form. Please copy the information from Part I, Block I to the *Attention block* on the envelope.

PART IV -- FINAL PROJECT REPORT -- SUMMARY DATA ON PROJECT PERSONNEL

(To be submitted to cognizant Program Officer upon completion of project)

The data requested below are important for the development of a statistical profile on the personnel supported by Federal grants. The information on this part is solicited in resonse to Public Law 99-383 and 42 USC 1885C. All information provided will be treated as confidential and will be safeguarded in accordance with the provisions of the Privacy Act of 1974. You should submit a single copy of this part with each final project report. However, submission of the requested information is not mandatory and is not a precondition of future award(s). Check the "Decline to Provide Information" box below if you do not wish to provide the nformation.

Please enter the numbers of individuals supported under this grant.
Do not enter information for individuals working less than 40 hours in any calendar year.

	Senior Staff		Post-Doctorals		Graduate Students		Under-Graduates		Other Participants[1]	
	Male	Fem.	Male	Fem.	Male	Fem.	Male	Fem.	Male	Fem.
A. Total, U.S. Citizens	2				3	1				
B. Total, Permanent Residents										
U.S. Citizens or Permanent Residents[2]:										
American Indian or Alaskan Native										
Asian............................										
Black, Not of Hispanic Origin.........										
Hispanic										
Pacific Islander										
White, Not of Hispanic Origin										
C. Total, Other Non-U.S. Citizens										
Specify Country 1. India					1					
2. P. R. China					1	1				
3.										
D. Total, All participants (A + B + C)	2				5	2				
Disabled[3]										

☐ Decline to Provide Information: Check box if you do not wish to provide this information (you are still required to return this page along with Parts I-III).

[1] Category includes, for example, college and precollege teachers, conference and workshop participants.

[2] Use the category that best describes the ethnic/racial status fo all U.S. Citizens and Non-citizens with Permanent Residency. *(If more than one category applies, use the one category that most closely reflects the person's recognition in the community.)*

[3] A person having a physical or mental impairment that substantially limits one or more major life activities; who has a record of such impairment; or who is regarded as having such impairment. *(Disabled individuals also should be counted under the appropriate ethnic/racial group unless they are classified as "Other Non-U.S. Citizens.")*

AMERICAN INDIAN OR ALASKAN NATIVE: A person having origins in any of the original peoples of North America and who maintains cultural identification through tribal affiliation or community recognition.

ASIAN: A person having origins in any of the original peoples of East Asia, Southeast Asia or the Indian subcontinent. This area includes, for example, China, India, Indonesia, Japan, Korea and Vietnam.

BLACK, NOT OF HISPANIC ORIGIN: A person having origins in any of the black racial groups of Africa.

HISPANIC: A person of Mexican, Puerto Rican, Cuban, Central or South American or other Spanish culture or origin, regardless of race.

PACIFIC ISLANDER: A person having origins in any of the original peoples of Hawaii; the U.S. Pacific territories of Guam, American Samoa, and the Northern Marinas; the U.S. Trust Territory of Palau; the islands of Micronesia and Melanesia; or the Philippines.

WHITE, NOT OF HISPANIC ORIGIN: A person having origins in any of the original peoples of Europe, North Africa, or the Middle East.

NCR-9305115 – Multiprotocol Architectures as a Paradigm for Achieving Interoperability

Co-PIs: *Mostafa H. Ammar* and *Kenneth L. Calvert*

II. Summary of Completed Project

A significant obstacle to achieving heterogeneous system interconnection is the need to maintain compatible network protocols on all communicating systems. This is especially true as families of competing protocols proliferate, forcing consumers to choose and thereby risk inability to communicate with others who make different choices. This project has investigated and developed mechanisms and techniques that make it easier for systems to support and use multiple suites of protocols, and which make it easier to deploy new protocols on different platforms. Begining with a model of the relationships among communication protocols, the problem of making effective use of existing, installed protocols was investigated. Results include techniques for determining the degree of overlap in supported protocols, and methods of selecting a set of protocols to use from among the overlap.

The project has also compared and evaluated the specialized software environments provided by operating systems to support protocol implementations, and developed tools that enable protocols to communicate across different environments within the same operating system. In addition, encapsulation techniques have been demonstrated, which enable a protocol developed within one environment to be used in a different environment. Finally, insights developed in this project have been fed into the design of a next-generation protocol framework, which is explicitly designed to reduce the cost of achieving interoperability.

III. Technical Information

Students supported on this project include: Russ Clark (PhD, May 1995) Richard Clayton, Tianji Jiang, Bobby Krupczak (expected to graduate Summer 1997), Robin Kravets, and Xue Li.

In addition to the research papers listed below, the project has produced implementations of the various tools, protocols and auxiliary components used and/or developed as part of the work. Among these are:

- Implementations of the AppleTalk protocol suite in the BSD, Streams, and x-kernel subsystems.
- A set of protocol adaptors for interfacing Streams protocols to BSD protocols and vice versa.
- Encapsulation code for importing Streams protocol modules into BSD and vice versa.

All of the code resulting from the project is available on the World-Wide Web at:

`http://www.cc.gatech.edu/computing/Telecomm/playground/`

Publications

1. Calvert, K.L., "Beyond Layering: Modularity Considerations for Protocol Architectures", Proceedings of International Conference on Network Protocols, October 1993, pp. 90–97.
2. Clark, R. J., Ammar, M. H., Calvert, K. L., "Multiprotocol Architectures as a Paradigm for Achieving Interoperability", Proceedings of INFOCOM 93, March/April 1993, pp. 1342–1349.

3. Clark, R. J., Calvert, K. L., Ammar, M. H., "On the Use of Directory Services to Support Multiprotocol Interoperability", Proceedings of INFOCOM 94, June 1994, pp. 784–791.

4. Clark, R. J., Ammar, M. H., Calvert, K. L., "Multiprotocol Interoperability in IPng", RFC 1614, June 1994.

5. Clark, R. J., *Solutions for Ubiquitous Information Services: Multiple Protocols and Scalable Servers*, Ph.D. Dissertation, College of Computing, Georgia Institute of Technology, May 1995.

6. Clark, R. J., Ammar, M. H., Calvert, K. L., "Protocol Discovery in Multiprotocol Networks", Proceedings of Intl. Conf. on Comp. Comm. Networks 95, September 1995, pp. 361–368.

7. Talpade, R., Ammar, M. H., "Single Connection Emulation (SCE): An Architecture for Providing a Reliable Multicast Transport Service", Proceedings of the IEEE International Conference on Distributed Computing Systems, Vancouver, B.C., Canada, June 1995.

8. Clark, R. J., Ammar, M. H., "Providing Scalable Web Service Using Multicast Communication", Proceedings of the Workshop on Services in Distributed and Networked Environments, Whistler, Canada, June, 1995.

9. Clayton, R., Calvert, K. L., "Structuring Protocols with Data Streams", *Journal of Electrical Engineering Australia*, 16(1), March 1996, pp 29–36.

10. Krupczak, R. D., Ammar, M. H., Calvert, K. L., "Multi-Subsystem Protocol Architectures: Motivation and Experience with an Adapter-Based Approach", Georgia Institute of Technology, College of Computing Technical Report GIT-CC-95-08, February 1995. (Abstract presented at the Workshop on High Performance Communication Systems, Mystic, Connecticut, October 1995.)

11. Krupczak, R. D., Ammar, M. H., Calvert, K. L., "Multi-Subsystem Protocol Architectures: Motivation and Experience with an Adapter-Based Approach", Proceedings of INFOCOM 96, March 1996, pp. 1149–1156.

12. Cheung, S. Y., Ammar, M. H., Li, X., "On the Use of Destination Set Grouping to improve Fairness in Multicast Video Distribution," Proceedings of IEEE INFOCOM '96 Conference, March 1996, San Francisco, CA, pp 553–560.

13. Krupczak, R. D, Calvert, K. L., Ammar, M. H., "Protocol Portability Through Module Encapsulation", College of Computing, Proceedings 1996 IEEE International Conference on Network Protocols, Columbus, Ohio, October 1996. (Also Technical Report GIT-CC-96-12, Georgia Tech College of Computing, April 1996.)

14. Cheung, S. Y., Ammar, M. H., "Using Destination Set Grouping to Improve the Performance of Window-Controlled Multi-Point Connections," *Computer Communications,* Vol. 19, 1996, pp723-736.

15. Krupczak, R. D., Calvert, K. L., Ammar, M. H., "Improving the Portability and Reusability of Protocol Code", to appear in *IEEE/ACM Transactions on Networking.*

16. Clark, R. J., Ammar, M. H., Calvert, K. L., "Protocol Discovery in Multiprotocol Networks", to appear in *ACM Wireless Networks.*