June 23, 1992
Expiration Date: December 23, 1992
Daisy Shen
daisy@watson.ibm.com

IBM T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, N.Y. 10598

TABLE OF CONTENTS

NETWORK DATABASE PROTOCOL BY DAISY SHEN

STATUS OF THIS MEMO

This document is an Internet Draft. Internet Drafts are working documents of the Internet Engineering Task Force (IETF), its Areas, and its Working Groups. Note that other groups may also distribute working documents as Internet Drafts.

Internet Drafts are draft documents valid for a maximum of six months. Internet Drafts may be updated, replaced, or obsoleted by other documents at any time. It is not appropriate to use Internet Drafts as reference material or to cite them other than as a "working draft" or "work in progress."

Please check the I-D abstract listing contained in each Internet Draft directory to learn the current status of this or any other Internet Draft.

This memo provides information for the Internet community. It does not specify any Internet standard. Distribution of this memo is unlimited. Please send comments to ietf-ndb@ucdavis.edu.

ABSTRACT

This memo defines a protocol for use with relational database systems in a TCP/IP based internet environment. This protocol is intended to make interoperability among different database systems possible. It is built on RPC (Remote Procedure Call) with a client/server type of relationship. This memo also defines the concept of Unit of Work. The work of Network Database is involved with Data Conversion, Security, I/O Buffer Management, and Transaction Management. They will be described one by one in this document.

OBJECTIVES

The Network Database protocol has three objectives:

1. To allow any workstation/mainframe users and applications to issue SQL statements interactively or imbed SQL statements within the application program, such as a SELECT statement, to access any database with any operating system.

2. To distribute database host machine's applications to the requester machine which is an independent workstation or a mainframe. Some requests need a lot of CPU time or memory, but some don't. The Network Database will make the requester machine more capable of handling some of the work load for the host where the database resides, unless the request needs more CPU time and memory than requester machines can handle. Therefore, a task can be distributed. Part of it is processed on the requester machine, and part of it is processed on the server machine where the database is.

3. Both the server and the client can be any workstations or mainframes. This is the prospect of interoperability.

OVERVIEW

This is a model of a client/server type relationship. There are two parts involved. One is the client, and the other is the server. The client is the one who issues a request, and the server is the one who provides the service for the request. The client machine can be called the requester machine, and the server machine can be called the database machine. In spite of the differences among all hardware, operating systems, and databases, there should be a piece of software to make them all communicate among themselves on accessing databases. That is what Network Database is all about.

Requirements

The NDB protocol is straight forward; it uses the existing standard relational database systems. There is no need to change the design, structure or implementation of existing relational database systems. However, it does have some requirements. The requirements are listed below:

TCP/IP

be installed. NDB protocol uses TCP/IP as its communication's protocol. NDB builds client and server code on top of TCP/IP. TCP/IP has been standardized.

RPC For both client and server machines, RPC should be installed. At the moment, there is no standard RPC implementation. However, NDB protocol is not restricted to any RPC implementation. Once RPC is standardized, NDB will then use the standard RPC. Before it happens, any implementation of RPC can be used by NDB.

Security There must be a means for security. However, it is not restricted. The common one is Kerboros. If either client or server machine does not have Kerboros installed, password checking or other security system should be used.

Data Conversion Abstract Syntax Notation(ASN) with Basic Encoding Rules(BER) is an ISO standard for data rep-

resentation and data conversion. It has been used widely. NDB adopted this standard for its data representation and data conversion. If the implenetor does not want to use ASN/BER, he/she may use a similar format as ASN does. It is shown below:

Format of data representation

Tag		Actual Value
Data type	Length	Value

Data Structure There is a control block passing between a client and a server. It contains required information for both client and server machine. The detailed information will be discussed later.

Standard Relational Database Since NDB is a network database system, a relational database system is required. NDB can be used for all standard relational database systems. Although these relational database systems are standard, their implementation can be varied.

SQL preprocessor All standard relational database systems have a standard procedure which is SQL preprocessor. Since each database has its own implementation, each SQL preprocessor of its database system is implemented differently. All components listed below should be all portable except the DB Utility. DB Utility is a component which performs SQL preprocessing. Although implementing DB Utility varies somewhat, its basic functions, procedures, and structures should be the same.

DESIGN

The Client Machine

Clients can be any mainframe or any workstation where end users submit requests. The components of a client machine are an End-User application, a NetDB client, and an RPC client.

End-User Applications

There are two kinds of end-user application. One is application programs that run on the client machine, and imbed SQL statements to access a remote database machine. The other is to issue SQL statements from the client machine interactively to access a remote database machine. The client machine runs the requests as its own tasks.

Imbed SQL statements in an Application Program
: can write an application program such as C, COBOL, FORTRAN, PL/I, PASCAL...etc., and imbed SQL statements in it. SQL statements will be processed the same way as interactive SQL commands.

Interactive SQL Commands An end user can type in a SQL command directly from the client machine to query or access data from a database such as Oracle or Ingres.

NetDB Client

The NetDB Client receives requests from the end-user application, parses the input arguments, and sets up a Unit of

Work by its status. It creates a thread of communication for the client. It packages the requests into a standard format which is called NDBC main control block, specifies the name of database to be connected, sets up an I/O buffer, and issues RPC calls to deliver the requests to the network.

The RPC Client

Although there is no industrial standard RPC, both SunMicrosystems' and APOLO/HP's implementations have been widely used. Many other vendors have ported one of these versions to their systems. If there will be a standard RPC, NetDB will use it. Otherwise, different implementations of RPC is not a major issue for the NetDB protocol. The RPC client performs a function call which is delivered to a remote machine; therefore, it is used by this protocol to deliver the calls from the client machine to the server machine. The RPC Client handles all the complicated network management work such as creating a socket, connecting sockets, maintaining a thread, sending and receiving data through a socket...etc. All those above functions are transparent to rpc users and end users.

When the NetDB Client finishes packaging the request and issues a procedure call which sends the request to the server, it uses the RPC protocol. It is the module which manages RPC calls and delivers calls through TCP/IP to the destination server machine. RPC uses the client/server model, and is designed for the support of network applications. Using RPC avoids the details of interfacing to the network, and provides network services without requiring any underlying network. The call is transparent. The caller is not explicitly aware of RPC. The call is like a system call in C. It is independent of transport protocols. If RPC is running on top of TCP/IP; most of the work of reliable transport is done. But the application still needs time-outs and reconnection to handle server crashes. If RPC is running on top of UDP, the application must implement its own retransmission and time-out policy. NetDB can run on top of TCP and UDP. It is one of the parameters that users have to specify. The default value is TCP.

The Server Machine

The server machine is a machine where the database system resides. The server performs all the SQL requests were sent

from clients. Many issues have to be solved such as security, data conversion, data buffer management, transaction management...etc. There are four components involved in the RPC server. They are the RPC server, the Network DataBase server (NetDB server), the DataBase Utility (DB Utility), and the Database.

The RPC Server

The RPC server works hand in hand with the RPC client, and performs the other half of the functions of creating a socket, connecting sockets, maintaining a socket...etc. as described in the session of the RPC client. When the server machine receives an RPC from a client machine, its RPC server will interpret it, verify it, and distribute it to the NetDB server.

The RPC Server process is dormant awaiting the arrival of a call message. When one arrives, the server process extracts the procedure's parameters, computes the results, sends a reply message, and awaits the next call message. Each RPC must communicate with a dedicated port. As part of its initialization, a server program calls its host's PORTMAP to create a portmap entry for its program and version number. A server machine is required to handle many clients. The

TCP server keeps state of each state for each open client's connection; therefore, the number of clients is limited by the machine resources. The UDP server does not keep any state about the client, it can handle unlimited clients.

The Portmap Manager Server

The Portmap Manager Server is a server which manages all the program numbers that NetDB servers use. It initializes a table which contains information of each NetDB server machine's status. It is responsible for updating the status when it receives a request from its client. The Portmap Manager Server only has two kind of clients. One is NetDB clients and the other one is NetDB servers. The Portmap Manager Server does not need a well-known port. The port number is allocated by the portmapper during run time.

The NetDB Server

The NetDB server receives the main control block (NDBC) which was created by the NetDB Client and was passed from the NetDB Client to here. The NetDB server will verify all

the items in the control block, and use the information to set up the transaction manager table. It also performs transaction management. The NetDB Server does not need a well-known port. The port number is allocated by the portmapper during run time.

Transaction Manager Most databases perform single thread although there are a few databases that can perform multiple threads. It works well with one to one client/server relationship. NetDB wants to make a server machine work with multiple client machines. It causes a problem for the server to manage different requests from multiple clients simultaneously. Although clients send requests to the server machine simultaneously, the server has to process them one by one. In order for the server to manage multiple threads, the concept of a Unit of Work is created, and the server machine CONNECT as other user is used. The transaction manager performs the function of managing Unit of Work.

Unit Of Work(UOW) Unit of Work is a mark for the server to recognize the threads which came from clients. A client starts a UOW by issuing a begin command. The client's database machine's user id and and its password will be requested. The client has to type in both; however, the password will not be shown on the screen. After begin, the

client can issue as many SQL statements as it wants. The client issues an end command to end the UOW. Only the first request in UOW will be prompted for database machine's userid and its password. Both pieces of information will be stored after the first request and destroyed when the last request is issued.

If a client only wants to send one SQL request, the client doesn't have to start a UOW. The client simply sends one request, and the server treats the request as an entire UOW. The database machine's userid and its password are required.

Name Mapping For security, Kerberos must be used to authenticate that the client is really who he claims. For those database machines where Kerberos is not ready, the name mapping is necessary to the Network Database. Usually a database only recognizes userids on its machine; therefore, mapping clients' machine ids and userids to the database host's userids has to be done.

When the NetDB server receives a SQL request, before it

hands the request to the database system it has to perform the name mapping. The function only needs to be done at the beginning of each Unit of Work. The first request of a Unit of Work has to append a password of its database host machine's userid so that the NetDB server can do authentication.

The NetDB server goes to the name mapping file/database to match the corresponding machine id and userid. It then issues a call to the database host to verify if the userid and the password are matched. If the verification is positive, the server continues to perform the request. Otherwise, the request is rejected.

Data Conversion One of the advantages of RPC is that the RPC application does not have to do any data conversion. RPC uses its own data conversion routine to convert different data representations between different computer architectures. However, NetDB cannot take this advantage. The main reason is that to use RPC's data conversion, the application has to give the data type as one of the arguments for RPC to do the conversion when the call is made. In NDB's case, most of the time, the requester doesn't have the knowledge of the data type. Therefore, the NetDB client cannot provide this information when the request is made. The NetDB server has to do data conversion for transferring data between different machines. One of the ways is to use ASN.1.

ASN.1 is Abstract Syntax Notation One (ASN.1). It defines the formats of the data exchanged by different machines. NetDB will use it as the data representation for data conversion between machines. The NetDB server packages the data back from database into ASN.1 format. Then NetDB can

do the data conversion according to ASN.1 format and its rules.

Multiple Databases

The NetDB server supports multiple databases on a host machine so that the end user can specify which database to connect when he/she submits the request.

The DB Utility

A true implementation of an ANSI standard relational database system must have a set of SQL preprocess procedures for its application. The NetDB server is database's applica-

tion. The preprocess procedure is unavoidable. The DB Utility performs these procedures. Each SQL statement has to go through the SQL preprocess. Although the implementation varies among different machines, there is an ANSI standard to follow. Oracle, Ingres, Sybase, Informix...etc. all follow the ANSI standard. For instance: Oracle has a set of procedures named precompiler which contains functions such as open, connect, prepare, describe disconnect, close...etc. for processing either static or dynamic SQL statements. Other venders may call it preprocessor, but they all serve the same purpose.

The DB Utility simplifies the database access process for applications and users. A general SQL preprocess procedures include CONNECT, PREPARE, DESCRIBE, OPEN, FETCH , CLOSE and DISCONNECT. DB Utility provides processing all these functions. DB Utility also provides I/O buffer management.

One of the functions is CONNECT which allows application programs to connect to a database, and use it. It provides a 'Call' interface to the database connection services. Application programs can control the exact state of their connection to the database. It is restricted to a single task level. It is the DB Utility's responsibility to understand and manage the connection status of multiple tasks in single database system.

Usually there are two ways to connect to a database system. One is explicit connection which uses the preprocess calls, such as CONNECT, OPEN, CLOSE, DISCONNECT. The other is implicit connection which doesn't use the above functions, but starts making SQL calls. In other words, if an SQL call occurs before the connection to the database has been established, the connection to the database is implicit;

otherwise, the connection is explicit. Most users will use the explicit connection services to maximize the flexibility and control.

Internal Procedures of SQL Preprocess for a Select Statement

CONNECT Establish a connection between the current address space and a database (if it is not already done), and establish the current task as a user of a database services.

OPEN Establish the current task as a user of a database

services(if it is not already done), and allocate database resources for SQL access(create a thread).

SQL Calls SQL calls pass through the Language Interface by branching to the entry point. Usually it is a FETCH.

CLOSE Deallocate database resources(terminates the thread), and remove the task as a user of database services if the connection was established by OPEN instead of CONNECT.

DISCONNECT Remove the task as a user of a database services, and terminate the address space connection to a database if this is the last(or only remaining) task in the address in the address space established as a user of a database services.

Database

The database machine can be any ANSI standard relational database system, such as Oracle, Ingres, Sybase, Informix, IBM SQL/DS, IBM DB2, and the Dec's database. They process the valid SQL request delivered from the NetDB server, and return the result back to the NetDB server.

Security

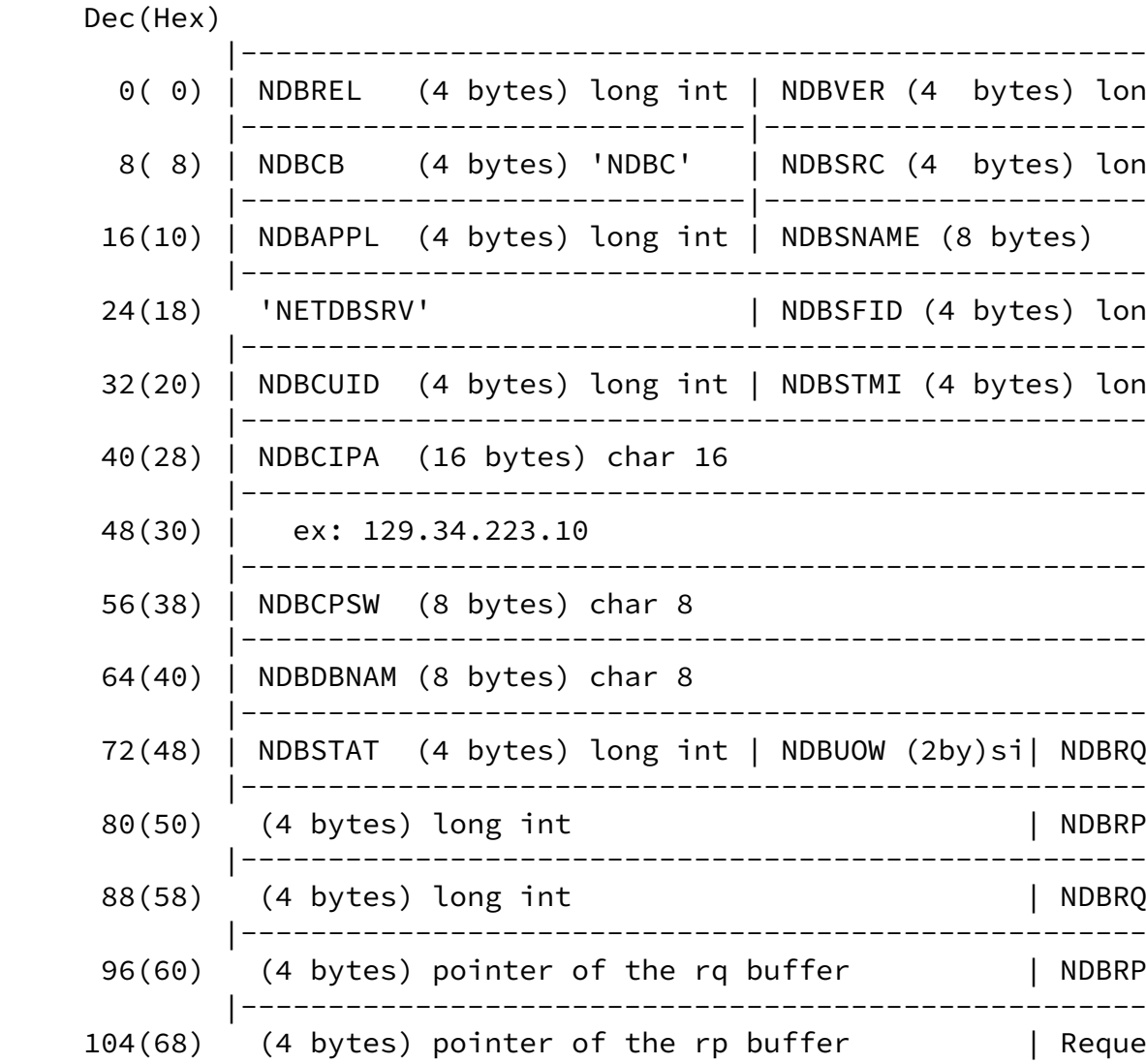
There are two levels of security. First is the network security. The Network verifies that the clients are authorized to get through the network. Kerberos is a known security authentication system in Unix. Second is the database security. The database verifies that users are authorized to access the database. The database system administrator authorizes the different attributes to the

different userids. For example: some users can read and write certain tables, but some users can only read them; some other users may have even more restrictions that they can only access certain fields of particular tables. That protects the database such as DB2 or SQL/DS. The NetDB relies on these two levels checking as its security system.

The Main Control Block, NDBC

The most important data structure in Network DataBase protocol is the main control block which is called NDBC. The NetDB Client has to get a piece of storage for NDBC and its I/O buffer. The format of NDBC is shown below:

The Main Control Block, NDBC, is shown below:



```
            |-------------------------------------------------
    112(70) | Buffer ( variable length, v1)
            | ....
            |-------------------------------------------------
    112+v1  | Reply Buffer ( variable length, v2)
            | ...
            |-------------------------------------------------
```

```
    NDBREL:   (l int)   Release number
                        x: release x, where x is an integer
    NDBVER:   (l int)   Version number
                        y: version y, where y is an integer
    NDBCB:    (char 4)  Self Identification.  It is always 'NDBC'
    NDBSRC:   (l int)   Server Return Code
    NDBAPPL:  (l int)   What kind of application, C, assembly, or
                        1: C   2: Assembly 3: Others
    NDBSNAME: (char 8)  The Server Name
                        'NETDBSRV' for NETwork DataBase Server
    NDBSFID:  (l int)   Service Function ID
                        1: begin, 2: end, 3: select
    NDBCUID:  (l int)   Client Uid on the database host's
    NDBSTMI:  (l int)   Server Transaction Manager's Index
    NDBCIPA:  (char 16) Client IP address
    NDBCPSW:  (char 8)  Client's Password for the database host m
                        Only used in the first request of a UOW.
```



```
    NDBDBNAM: (char 8)  The name of th database that the end user
                        to be connected.
    NDBSTAT:  (l int)   The status of the current thread
                        0: only one request in the thread
                        1: begin, 2: end, 3: in the middle
    NDBUOW:   (s int)   Number of Unit Of Work (Number of Thread)
    NDBRQDLN: (l int)   The length of Request Data Buffer
    NDBRPDLN: (l int)   The length of Reply Data Buffer
    NDBRQD:   (char< >) Request Data Buffer, variable length
    NDBRPD:   (char< >) Reply Data Buffer, variable length
```

Implementation

For more information about implementation of NetDB, please refer to Network Database Implementation Information Internet Draft.

Operational Procedures

1. Bring up the Portmap Manager Server

2. Bring up the NetDB Server. It can be replicated for multiple times if users need multiple NetDB servers. NetDB servers have to be brought up after the Portmap Manager Server has been brought up.

3. Once above server are all brought, end users can start submitting requests.

COMPONENTS IN THE NETWORK DATABASE PROTOCOL

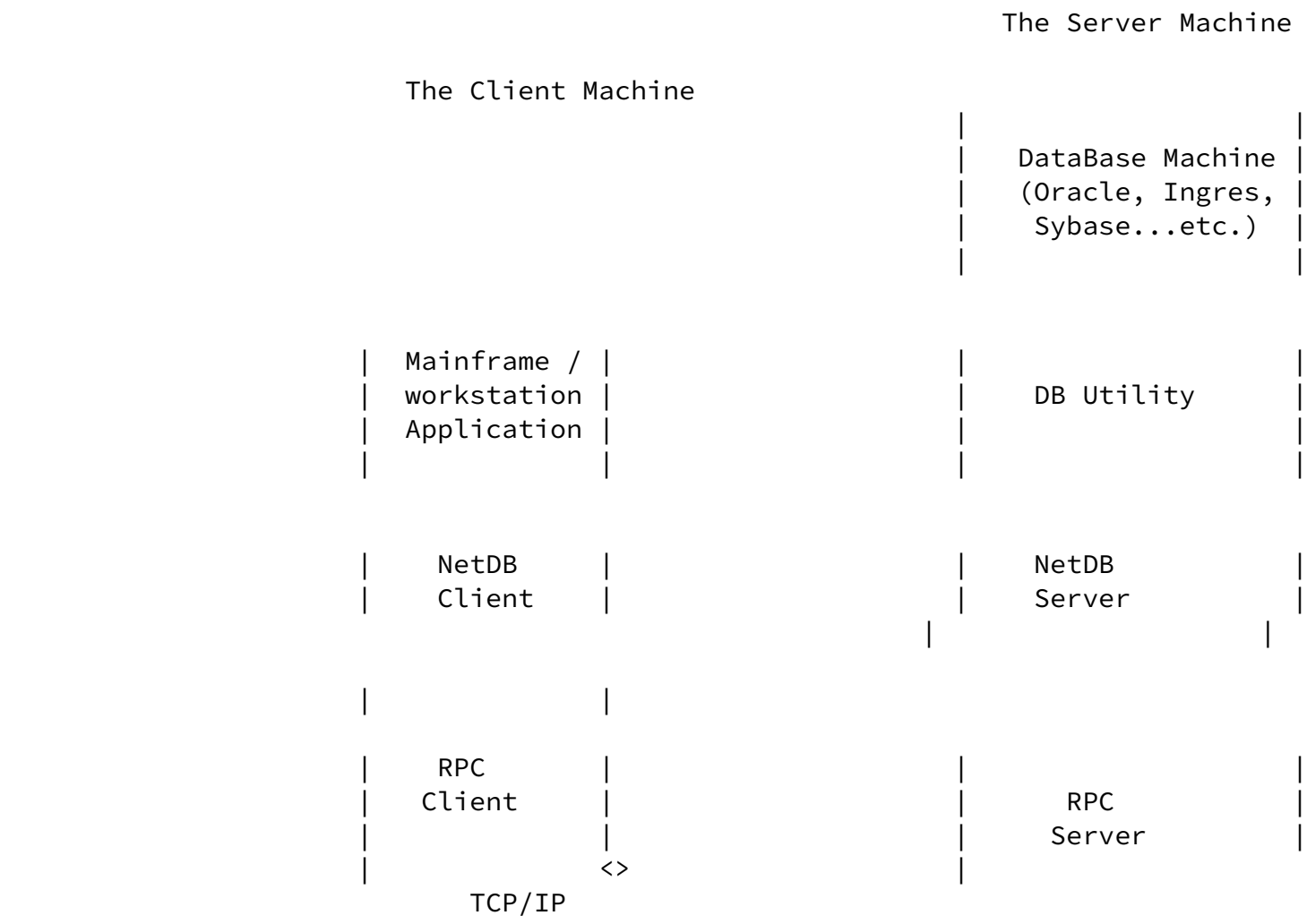
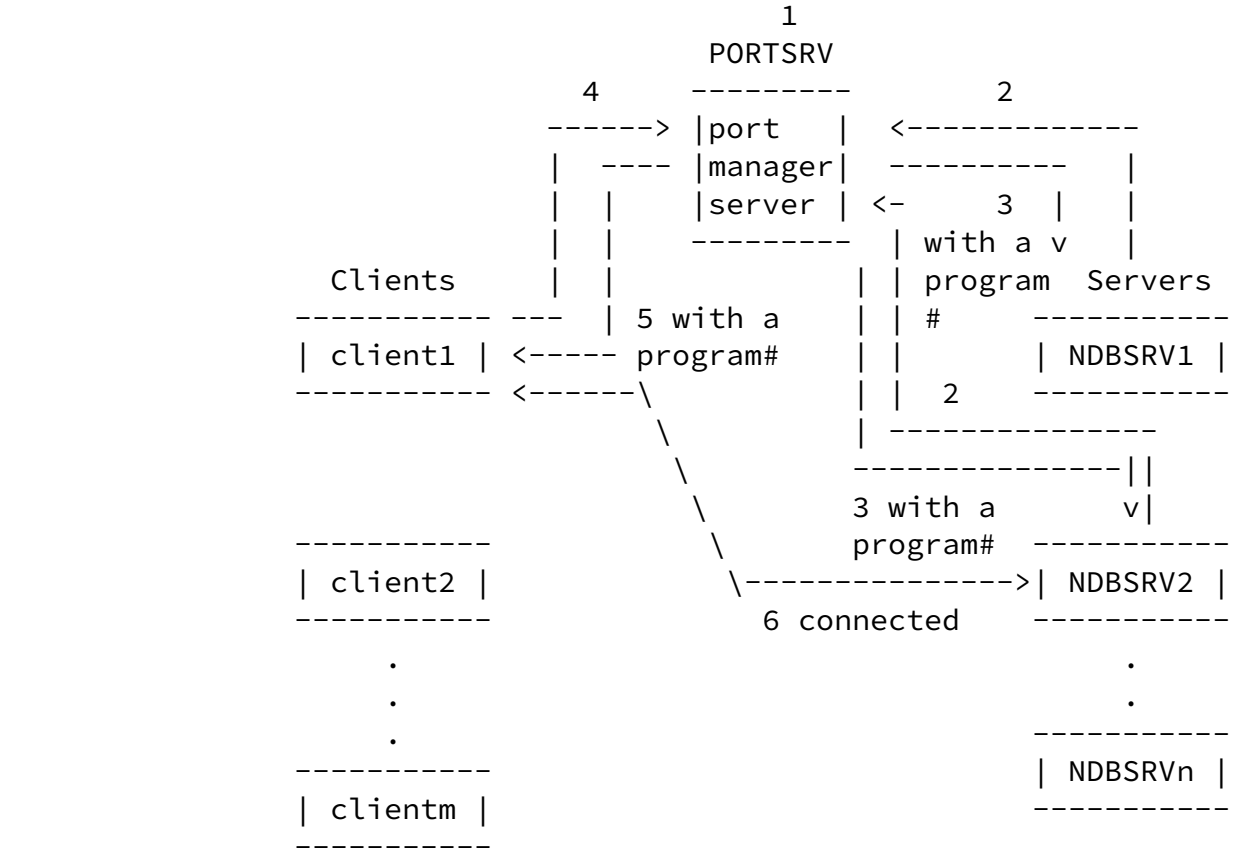


Figure 1. Components in the Network DataBase Protocol

RELATIONSHIP AMONG PORTMAP MANAGER SERVER AND NETDB SERVERS



Steps:

1. Bring up the PORTSRV

2. NDBSRVx issues a request to PORTSRV to get a program number

3. PORTSRV updates its program status and returns a program number

4. A client issues a request to PORTSRV to get a program number of an available port

5. PORTSRV returns an available program number

6. When the client issues a request from now on, they are connected