

Migration of Dissimilar Data Base and Application Servers

David Grothe

March 2010

Introduction

This white paper addresses the problem of accessing multiple dissimilar data base and application servers. Due to historical evolution of server sites, their network access methods and multiple mergers and consolidations it is possible that a single organization finds itself having to manage and utilize multiple application and data base servers with nothing in common among them., *except their user community.*

The focus of this paper is on the migration steps that *you* can take to integrate these dissimilar servers into a more coherent system so that users of the various servers can access them in a simpler and more effective manner.

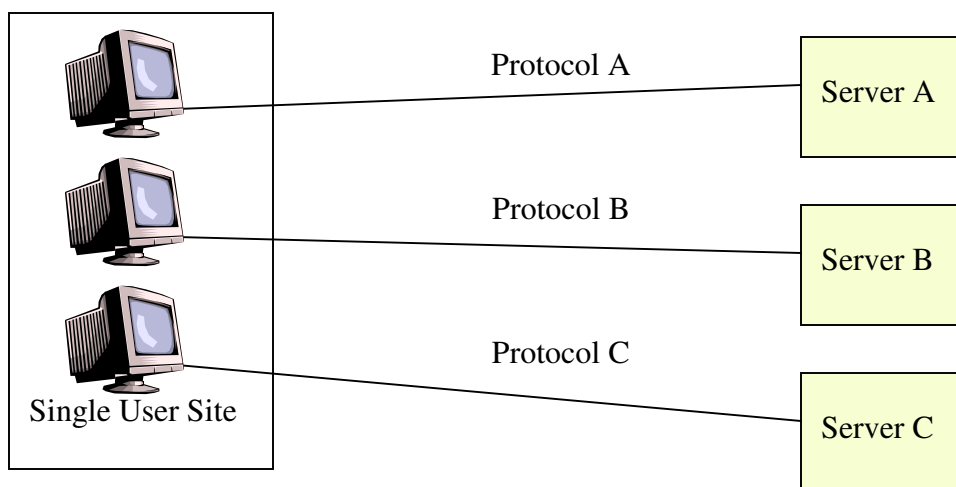
Simpler, in that the confusing and redundant networking that usually accompanies such configurations can be rationalized into a single IP based network.

More effective, in that data base queries that once had to be submitted by hand to multiple servers can now be submitted to all or some subset of the servers automatically.

We propose a set of migration steps that one can perform that can lead to short term increases in the productiveness of these servers and over the long term facilitate the transition to state of the art networked access to multiple servers.

The Problem

The problem is illustrated by the following drawing. We have depicted three data base/application servers each with its own individual network access. In the extreme this leads to users of these three systems having three terminals on their desks, each terminal connected to one of the three servers. Real world systems such as depicted here involve a substantially larger number of servers.



In this example each user site has three physical terminals, one for each DB/Application server. Potentially, each server has a different communications protocol that must be used for submission of queries or other interactions with the server. Potentially, there are three separate network connections at each user site. In some cases these network connections could be old leased line synchronous links to the servers.

Imagine that there are hundreds or thousands of user sites such as this and the problem comes into focus. There may be hundreds of leased lines and hundreds of ports on each of the three servers to accommodate all the users.

Any single user cannot submit a query to all three servers except by formulating the query separately for each server in that server's special vocabulary. Correlation of results must be done by hand since the three terminals are physically separate and different. With luck one or more of the "terminals" might be a terminal emulator running on a computer so that results might be copied and pasted into some common document, with all of the associated risks of data loss or transcription errors.

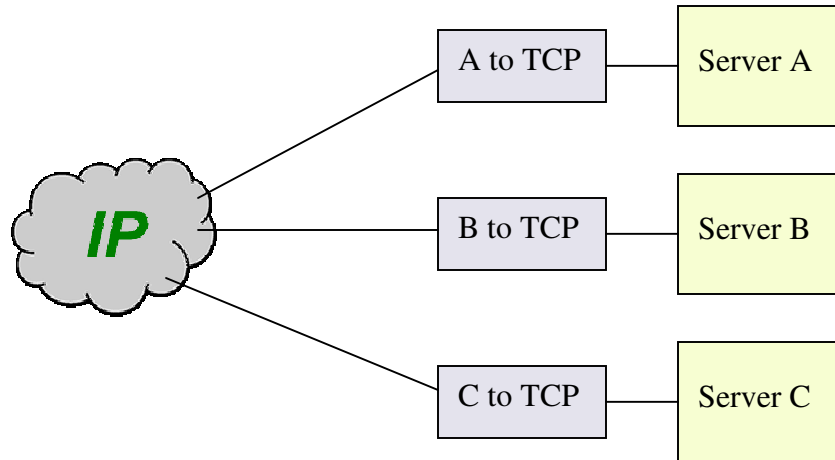
How to Unify Such a Configuration

Clearly there is motivation to unify system configurations such as this. The steps needed to accomplish this are conceptually straightforward. These steps are:

- Unify the communications protocols
- Eliminate the terminals
- Choose a Standard Query Protocol
- Create a Meta Server
- Add New Servers
- Replace Old Servers

Unify the Communications Protocols

First, the different communications protocols need to be unified. The obvious way to do that is to translate each protocol into TCP/IP. This is accomplished by placing a protocol translator in front of each server. The translator implements each protocol A, B and C. It then extracts the payload data from these protocols and sends that payload towards the user using TCP connections. This is illustrated as follows.



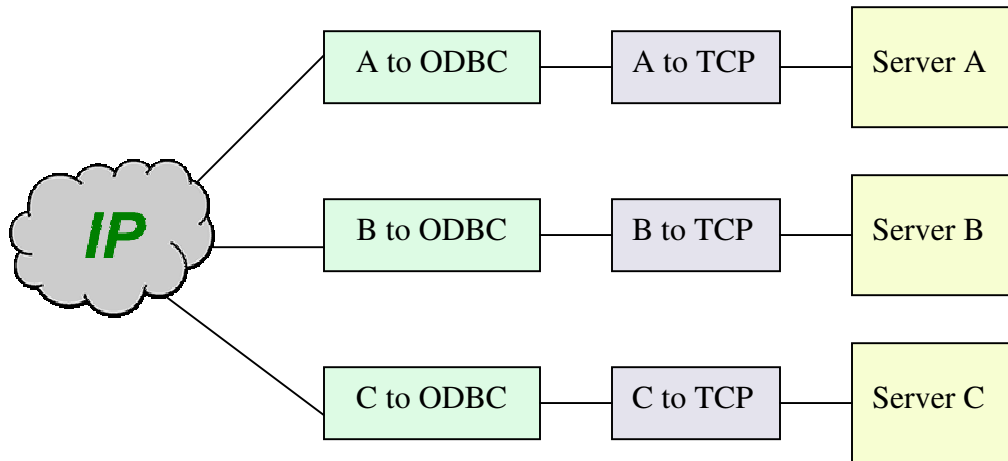
Eliminate the Terminals

The next logical step is to eliminate the legacy terminals at the user sites. Since the protocols A, B and C no longer extend to the user sites these terminals no longer serve any useful purpose. The logical choice of replacement is a PC or some platform that allows browser access to the IP network.

Choose a Standard Query Protocol

Select a standard data base query protocol that is defined over TCP connections. The idea is that all data base queries will be formed in this protocol. The obvious candidate for this protocol is ODBC.

Once this has been decided upon, develop three protocol translators for the servers A, B and C. Now the data from these three hosts is in a standard format and is communicated to the user using TCP connections.



It is our idea to perform the query translation in a separate hardware device from the protocol translation. The protocol translation from A, B or C to TCP can be performed in a commercial off the shelf product. The query translation software must be developed specifically for each server and individual data base query formats and/or application message formats.

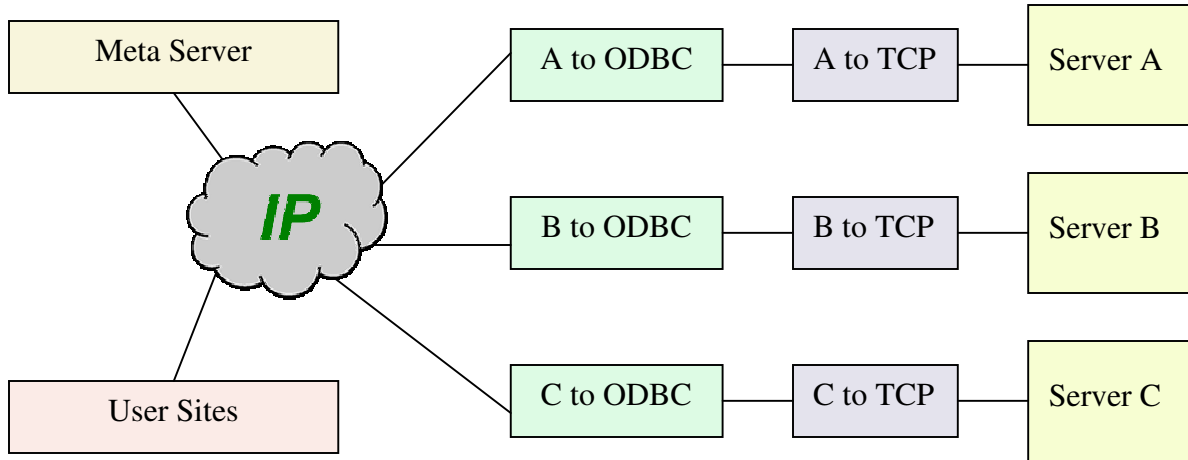
Developing these message query format translators is work of significant magnitude. However, it is assumed that this translation involves much less work and less expenditure of funds than replacing the applications on the servers, or replacing the servers themselves.

Create a Meta Server

What is needed next is a new server that can submit automated queries to the three servers. The meta server communicates with the user via the IP network. It receives data base queries or application interaction messages in the standard ODBC message format. It then forwards these messages to the individual servers via the message translators.

The algorithms in the meta server might take the form of Java applications that mimic the operations of the server applications. The Java app obtains its data from the servers and interacts with the user. In the simplest case it is simply mapping transactions to screens.

However, the meta server can add value to the system. There can be Java applications that interact with multiple servers in order to service individual user requests. An example of this would be querying all three data bases for a common search term. The meta server can issue the queries, await responses and then correlate the responses for presentation to the user. Having the meta server in place opens the possibility of much more sophisticated use of the multiple data base and application servers.



Add New Servers

At this point you can add new servers that already use ODBC/TCP. These servers can be programmed simply for their back end functions with the user interface code being developed for the meta server. New servers can then easily be added to the broadcast query mechanisms of the meta server.

Replace Old Servers

When convenient, in terms of timing and budget, you can replace the older servers with new equipment and/or new application software that integrates into your IP based network. The meta server still retains its value in distributing and managing queries to server back ends.

User's now have a common interface and can be completely insulated from changes in server technology. Thus the users need to be retrained only once in the use of the meta server. After that no user training is necessary to change the server technology.

Gcom's Contribution

In the above plan, Gcom can provide the protocol translator that terminates the legacy protocols of SNA, LU6.2, X.25 or Bisync and forwards the resulting payload data in TCP packets to the query translators. It is a small part of a big job but it is essential to be able

to perform this step. API programming for legacy protocols is difficult and error prone and there is no sense in adding this source of error to an already ambitious undertaking.