Open Source
vs
Commercial
DCE
Client/Server Software
Development Technologies.

Tom Brennan.

A dissertation submitted in partial fulfilment of the requirements of the Higher Education and Training Awards Council for the degree of Bachelor of Science in Information Systems

Department of Information Technology
Limerick Institute of Technology

March 14th 2003

i. Abstract

Why?

- I felt that it was important to carry out the topic, as ever, the movement is growing daily regarding free, Open Source software, and outline the benefits it has to offer. Of course not all free software has such forward momentum or increases with popularity depending on the needs of the individuals themselves.

What?

- The main focus of the research was into whether Open Source or free software have such viable market penetration and in this case, Client/Server DCE Software Development technologies. This dissertation aims to fulfil the facts as necessary as per in Chapters 3 and 5.

How?

- By researching the internet and downloading the Open Source or free software, and trying it out by implementing the project which will be outlined in Chapter 6.

Conclusions.

- The conclusions outlined, are entirely my own opinion, based from a software developer's viewpoint and also from an end-user's viewpoint. The balance between two ends of the spectrum will be outlined in Chapter 7.

ii. Acknowledgments.

Thanks Ita Kavanagh for giving me a grilling about my experience, when we first discussed about the dissertation when I started college back in October 2002. It was the grilling that gave me inspiration!

I would like to thank John Lannon for his useful input and constructive feedback, without him, this would never have got far. Thanks!

Thanks to Mark Carter, who gave me an insider's perspective on Entera.

Thanks to Phillip Ionio aka finieous, Wez Furlong aka wez, Bruce Foster for answering my inquisitive and silly questions regarding FreeDCE. finious and wez are the maintainers of FreeDCE.

Thanks to Denise Cloutier and Ray Cavanagh of Entegrity for their time in answering my queries regarding their evaluation software, despite lacking the requirements for the evaluation which I was unable to dig into.

Thanks to Jim Doyle for your work in porting OSF/DCE across to Linux, nonetheless an achievement on your part, yet drove me up the walls in attempting to get the build to work which proved futile! After those two days spent, my fingertips were well and truly worn down, known as finger-itis, as a result of chanting half-baked magic incantations on the command line!

One other person that deserves a thanks, Mr. Linus Torvalds, for your fantastic creation of Linux. Ever since I discovered Linux and Open Source back in 1994/5, it was a joy to play and hack around with, only for you, the "Linux Format" magazine wouldn't be here. 'Hail Linux - the king of the OS'!

iii. Contents.

iv. List of Tables.

v. List of Illustrations.

vi. List of Glossary Terms.

ActiveX – A recent software technology utilising Microsoft Windows platform in communicating with different objects commonly found in Visual Basic, such as ADO (ActiveX Data Object) which allows communication with the database.
ADSL – Asymmetric Digital Subscriber Line – a modified variant of DSL with capabilities of upload and download running at same bandwidth thereby maximising potential of digital lines.
API – Applications Programming Interface – a set of system calls that programmers can interface code with thereby maximising the power of code.
AIX – IBM's variant of Unix, based on System Five release (SysV)
Benefits – a benefit that constitutes part of an agreed health care insurance such as Dentistry, MHSA (see below), Maternal Care etc.
CSMA/CD – Carrier Sense Multiple Access with Collision Detection, a rule governing how information is passed using Ethernet technology.
CORBA – An acronym for **C**ommon **O**bject **R**equest **B**roker **A**rchitecture, an object orientated version of DCE where it's technology comes from.
CPU – Central Processing Unit, the heart of the computer, which performs complex operations.
DSL – Digital Subscriber Lines – a faster means of telecommunications, traditionally was analogue, now replaced by digital technology, the bandwidth for download is bigger than the bandwidth for upload hence always-connected is possible.
GNU – A recursive acronym, coined by the FSF, implies "GNU is Not Unix"
GPL – GNU Public Licence, a copy left licence, which allows software to be freely modified and distributed without cost.
Grep – An acronym originally taken from g/RE/p which stands for **G**lobal **R**egular **E**xpression **P**arser. A Unix command utility that can search through text for a certain sequence of characters.
FSF – Free Software Foundation, founded by Richard Stallman who promotes free software and distribute it under the GPL.
HMO – Health Maintenance Organization – A health insurance in which members pay a premium for services, often, members who choose to visit their non-network provider will often don't gain any benefits.
IDL – Interface Definition Language, a set of rules used to govern how RPC stubs are to be generated.
In-Network – A certain mile radius where members can visit their provider, without incurring extra charges. See *Out-Of-Network*.
LAN – Local Area Network, commonly a private network in an organisation.
Linux – An Open Source variant of Unix, which is by far, the most popular Open Source project ever seen, widely portable across diverse platforms.
MAC – Media Access Control, a fixed address that is unique and identifies each network interface.
MHSA – Mental Health, Substance Abuse, this governs psychiatric care and in general drugs/alcohol abuse.
MIN – Managed Indemnity – this is where a visit to any doctor can be arranged. All individuals pool their out-of-pocket resources together, which offers generous benefits.
Network – Consists of a group of doctors, pharmacies, hospitals collaborating together, and is demographic based.

Option – An extra component that makes up part of the benefit as whole, for instance, Maternal Benefit, the extra option(s) are Pre-Natal Care, Post-Natal Care. Options such as the given example usually cost extra but gives extra peace of mind.
OSI – Open Systems Interconnection, an organisation that have established standards on protocols.
Out-Of-Network – A member who travels outside a certain mile radius of network in order to reach a certain provider, and costs are incurred and thus comes out of their pocket.
Package – A healthcare insurance set-up and agreed by employers and insurance companies, this usually consists of Plans, Benefits, Options and Networks. A simple example, for sake of comparison with Ireland's VHI healthcare system is 'Option Plan B'.
Plan – This is set up by the Insurance organisation that offers healthcare incentive as part of an employee's salary. A plan is made up of Network(s) and is agreed by the employer and insurance organisation in question.
PCP – Primary Care Physician – A doctor that is agreed to look after illnesses for individuals. If such individuals need specialist opinions, the PCP will refer them to the relevant specialist as per the network arrangement.
PPA/PPO – Preferred Provider Arrangement/Organisation – An organisation offering worker's compensation, transplant networks and other provider networks at flexible costs yet can cover for pay out-of-pocket costs.
Posix – A standard that applies to Unix platforms, the more Posix compliant it is, the more interoperable and portable the platform is.
RAM – Random Access Memory, volatile memory storage available when the power is switched on.
RPC – Remote Procedure Call, a routine that appears to be executed locally, when in fact it is executed remotely.
SQL – Structured Query Language, a standard for sending command strings to the database to manipulate the data.
TCO – Total Cost Ownership, a formula to work out the cost of such product on a yearly basis. It is not accurate enough to inform the cost, as it does not cover unforeseen factors.
TCP/IP – Transmission Control Protocol/Internet Protocol, the protocol, which has a series of rules governing electronic information, exchanges across networks.
Unix – An operating system that has been around for nearly three decades, invented by Dennis Ritchie and Brian Kernighan, who also invented the language 'C'. Today there are variants of Unix commonly based around two central standards namely, AT&T and Bell Lab's System V. Also, despite having different variants, the term *nix is represented to define a general Unix platform.
WAN – Wide Area Network, a means of linking up more than one LAN via telecommunication lines.

Chapter 1.

1.1. Introduction to Dissertation.

1.1.1. About the Dissertation.

In this day and age, there is a lot negativity generated in relation to Open Source. There are many individuals out there that have heard about Open Source, yet would not touch it, purely out of fear because of the following factors:

- Lack of support such as "If I download this free software, who will support it?" This is the most common question and an important one.
- Quality of software, there is a tendency for Open Source software to be more stable and have very responsive turn-around time regarding fixes/patches.
- Well-known security exploits commonly found in Unix utilities such as sendmail. That is not to say, that Unix is insecure, it is secure, but due to bad programming practices, exploits have been well documented such as buffer overflows. *Despite that, chances are the code would be cleaned up and fixed due to code being under scrutiny by thousands of programmers, in a short period.*

The points have illustrated above, shows that there is a perception of uncertainty. Yet now, with big companies willing to spend money to develop Open Source and give it back to the community, this fear will subside in the near future. The one primary drive behind Open Source is the cost factor, which is free to use, free to modify and distribute.

Consumers felt that it would be a waste of time in looking for fixes/patches, or even upgrading Open Source software. That explains why they prefer to stick to commercial solutions where there is support to back up in the case of unforeseen problems, and thus would restrict their choices with proprietary software.

With Open Source, rapidly gaining attention and companies are sitting up taking notice. They are beginning to realise that they do have a choice after all, i.e. proprietary/open source software.

By looking into Open Source, consumers will be able to break away from closed and proprietary solutions and thus have a freedom of choice and to be able to choose which software is suitable, and furthermore, the source code is usually supplied thereby allowing the luxury of tailoring the software to their specific needs.

There is extensive analysis of whether Open Source is a viable alternative in today's business. ***(Kozinski 2002*** and ***Drummond 2001)***

Also, since there are a large number of corporations who are uncertain about whether to migrate or maintain their existing legacy applications i.e. make it more up-to-date. This uncertainty is there for a number of reasons such as cost, time and development. Even then, the maintenance of such legacy applications would be awkward if it was decided that a more modern software approach was glued on to the existing code, such as CORBA, WebSphere, MQSeries etc.

The corporations would effectively lose out, since dependency on such legacy applications is high and thus some might say, these are the backbone of their IT infrastructure, as well as for their day-to-day business operations. By taking a risk to bolt on say, CORBA for instance, the stability of legacy applications could and may very well tumble downwards rapidly due to the original designs of these applications.

Furthermore, a lot of companies using such DCE technologies, realise that it would be unfeasible to continue paying money for support as well as "keeping it alive". By looking into Open Source, an alternative can be found despite having the very basic means of DCE technology. Look at ***FreeDCE (2002)*** for more information.

The dissertation aims to narrow the gap in due reflection of this topic, by comparing both spectrums and drawing up conclusions to assert if Open Source software is a viable alternative where cost issues can be minimised.

While more recent developments such as CORBA, ActiveX, to name but a few have superseded DCE technologies, DCE is considered to be the forefather of the technologies we have today. Sadly, today, DCE is dying out as big companies pull their weight in supporting such technologies, IBM's WebSphere & MQSeries, Microsoft's ActiveX etc., yet have their place among legacy systems and applications and soon will be consigned to gathering dust along with COBOL!

The dissertation gleans in on the case study, which provided the inspiration to investigate DCE technologies. In Chapter 3 and 4, we cover the case study, describe how Entera works, its problem(s) and solution(s). In Chapter 5, we explore the realm of Open Source and alternatives called OSF/DCE and FreeDCE, Chapter 6 covers the implementation of FreeDCE, and finally in Chapter 7, the conclusions.

Also, covered in this aspect of Open Source vs. commercial technologies, is in terms of software development regarding DCE. And conclusions and comparisons are drawn up between the two areas of software, on the one side of the spectrum is open source (FreeDCE) and the other side is commercial (Borland's Entera and Entegrity) respectively.

1.1.2. Why I chose this topic?

The primary reason behind this topic is quite simple; I have worked in the software sector for nearly five years, specifically in the Client/Server development on the Unix platform. The knowledge and experience that I have gained are highlighted in the case study as outlined in Chapter 3. I have and still do, always admired how much Unix has to offer, from its wide range of software right down to the nuts and bolts software that requires a certain level of technicality.

1.1.3. What are the benefit(s) from this topic?

- The main benefit, which I will get from carrying out the tasks involved in setting up this dissertation, is gaining new knowledge - technical and general based on my ability, in carrying out the research in this field.

- The other benefit is that individual(s), be they software development managers, IT managers etc., will endeavour to make a good choice in determining which product is right. Moreover, to learn something from it. And try to keep an open mind as honestly as they can, when it comes to the essential process of decision making in this regard in determining the factors and weighing up the pros and cons in this area.

I will start by introducing networks and protocols; various methods of client/server models will be highlighted also. I will try to steer clear of technicality in order to make this dissertation fit in with a wide audience who has absolutely no prior experience although a general understanding of programming is advantageous but not a pre-requisite.

Chapter 2.

2.1. Introduction to Networks and Protocols.

In this chapter, I will be looking into how networking plays a fundamental role in DCE Client/Server Technologies and is a driving factor behind it. As far as the advancements of networking goes, such as the new implementation of IP version 6 which is being rolled out, wireless networking. With the advent of current trends in development such as CORBA, to name but a few, this is consigning DCE to rest. Furthermore, due to the influx of so many packages appearing which supports CORBA et al, this would drive management to consider that since CORBA et al, is more popular than DCE. This in turn would force them to re-evaluate software development as such. To re-iterate the factors are:

- Advancing network technologies. *Would DCE or similar, be able to cope with newer networking protocols such as IPv6, Wireless networks etc, would recompiling the code using the more up-to-date binary code solve the upgrading of IPv4 to IPv6 while maintaining compatibility with IPv4?*
- Trends and attitudes towards software technologies, especially in management organisations.

2.1.1. Networks.

Networks are a means of being able to connect more than one computer together, and to share resources on it, be it files, software, printers, or any other hardware devices. Networks come in two main flavours and topology (how it is represented diagrammatically, such as star, ring, to name but a few). There are two types of networks - LAN (Local Area Network), and WAN (Wide Area Network). In this chapter, when I talk about data passed around. I refer to the smallest known quantity of data put together in order for computers to distinguish who is the intended recipient, the source or sender. It also contains other esoteric bits and pieces of data that constitutes as a packet.

LAN's are commonly found in small companies and is usually private - only the employees have access to it. WAN's are a more common means of allowing LAN's in different regions to communicate with each other, for instance, a LAN in a Limerick-based office can access the main office in Dublin which also has a LAN via WAN. WANs are made possible via telephone lines, digital subscriber lines (DSL), asymmetric digital subscriber lines (ADSL).
There are two different flavours, which I will give a brief summary of and is outlined.

1. **Token Ring Network.**

   This form of networking requires a special communication access protocol called token passing, (a simple classification of how data is organised, prior to be sent through the hardware interface - NIC, an acronym for network interface card). That data is passed around from one computer to the next is known as a token. Any computer that wishes to send a token must wait until it is free to forward it on to another computer via special statuses contained in the token. This is analogous to having a traffic light system, if the status is green, the computer can then forward the token, and likewise if the status is red, the computer must wait. As with all token ring networks, this simply means that the token is passed on to each computer even if the computer in question is not the destined recipient for that token and therefore passes it along. Strictly speaking, it does not actually pass it on, it actually duplicates the token, change the status to green, and pass it on. ***(Luce, 1989)***

2. **Ethernet Network.**

   This is by far, a more popular form of networking. Ethernet uses specific communications access protocol called CSMA/CD - Carrier Sense Multiple Access with Collision Detection. All communication lines used in this regard have a special signal called a carrier. The procedure works like this, if there is no carrier, the computer is free to transmit data or packet. Using the same analogy, as above, if there is no carrier i.e., the traffic light is green, then the computer can send a packet. If the traffic light is red, this means, "no-go zone", so the computer must wait for a short period before retrying to send the packet. The thing about this form of protocol is that two computers could come along and check if the "coast is clear" and start sending packets, what happens. A collision of packets can occur,

and when it does, the two computers must wait for a short, if not random, period and retry again. In each of the packet that is sent, the receiving computer, even if it is not the intended recipient, must check the packet to verify if the destination address matches their own. If not, it forwards it on to the next computer, like passing the parcel.

Now that we have covered the two flavours of networks, it is time to cover the network protocols, which is "the rules" in how data is transmitted from one LAN to another via WAN. It is worth mentioning that the rules apply in both cases of LAN's and WAN's.

2.1.2. Protocols.

There are a number of diverse protocols which concerns with how the data is represented before it is broken down into its smallest quantity before being sent down through the hardware circuitry, i.e. 1's and 0's. That is the binary makeup of how computers work at its best. There are various protocols such as TCP/IP, UDP, NetBios, Novell, etc, to name but a few.

The most common protocol is called TCP/IP (Transmission Control Protocol/Internet Protocol). This is the rudimentary protocol used across all kinds of networks including the Internet. Had TCP/IP not been invented, there would have been no Internet!

2.1.2.1. TCP/IP.

In short, the IP (Internet Protocol) governs how packets are routed through from one computer to the next i.e. "How do I know this packet is for me?" TCP governs how the packets are sequenced, providing flow and transmission logic to ensure large data transfers arrive in a particular order.

A group regulates TCP/IP or committee called the Open Systems Interconnection (OSI for short). The committee establishes standards in how communications can be achieved across different devices. OSI have established a model that consists of two

layers, and each of the layers, there is a sub-layer. Application Interface and Network Interface, is the main two layers. In the Application Interface Layer, there are three layers, four layers for the Network Interface respectively. See Table 2-1 for the OSI model.

As of currently right now, IP is known as version four. There is a new version that has been ratified, but not yet fully implemented, known as IP version 6. The reason behind this is that the addressing scheme is running out of space using its current incarnation, due to the explosion of the "Information Superhighway" in the last decade.

IP version 6 is currently being rolled out in small steps; it does have backward compatibility with IP version 4. With the newer version, put simply, it has more room to allocate addresses and thus will never run out. The other major thing with this version is security, IP version 4, and software that used its suite of this protocol, had its fair share of problems regarding security. With the influx of so many programmers (in some circles and often poor media exposure, they are often portrayed as “hackers”) due to the explosion on the Internet, have exploited far too many security problems with the current incarnation. See Table 2-2, for the detailed description of the IP packet.

2.1.3. Layout of OSI Model.

Application Layer.	***Layer 7***	***Application.*** User programs and operations. e.g. *File Transfer, Terminal Emulation, Email.*
	Layer 6	***Presentation.*** Data entry/display, interface transformation. e.g. *Encryption, EBCDIC conversion to ASCII.*
	Layer 5	***Session.*** Control of data exchange, administration. e.g. *Rules such as hand-shake protocol, maintain connection, X.225 protocol definition (ISO 8327).*
Network Layer.	***Layer 4.***	***Transport.*** The transparent transfer of data between sessions e.g. *X.224 (ISO 8073)*
	Layer 3.	***Network.*** The procedure to assemble and route packets across networks e.g. *X.25 (Well known packet layer protocol), ISDN interfaces.*
	Layer 2.	***Link.*** The initialisation of data flow, error recovery e.g. *IEEE 802.3, 802.4, 802.5*
	Layer 1.	***Physical.*** The hardware circuitary interfaces to the communication media and cables, e.g. *RS 232, RJ 45, UTP Category 5.*

Table 2-1. OSI Model *(Umar, 1993).*

2.1.4. Layout of IP Protocol.

Here is the actual layout of the IP packet, the figures in brackets beside each description indicates how much bits are stored. For instance, "Version" field holds four bits, maximum value is 15, or in binary, is $1111_2 = 15_{10}$.

[1.] Version (4)	[2.] IHL (4)	[3.] T.O.S (8)	[4.] Total Length (16)
[5.] Id. (16)			[6.] Flags (3) / [7.] Frag. Ofs (13)
[8.] TTL (8)	[9.] Protocol (8)	[10.] Checksum (16)	
[11.] Src IP Addr. (32)			
[12.] Dst. IP Addr. (32).			
[13.] Options (8)			[14.] Padding
[15.] Data.			

Table 2-2. Detailed IP Packet. ***(Davidson, 1988)***

1 Version - This indicates the version of the protocol in use, as of today, in its current incarnation, it is usually 5 or 0100_2 in binary.

2 Internet Header Length indicates length of header, so the beginning of the data can be found **if** options are present. If no options are available, this is normally 5 or 0110_2 in binary.

3 Time of Service, this holds flags and precedence or priority of the packet, e.g. Low, high or reliability.

4 Total Length - this is measured in octets, which includes the header and the data, since the field is 16 bits long; the maximum value is 65,535 octets.

5 Identification, integers value to identify fragments. Fragments, occurs when the data is too big to be transmitted so it is broken down into smaller packets. This value is unique (usually incremented) if fragmentation applies.

6 Flags - this controls the fragmentation of packets. This can contain either MF (More Flag, i.e. There's more packets coming in that makes up a fragmented packet) or DF (Do not Fragment, i.e., there's sufficient data that can fit within the normal limit).

7 Fragment Offset - This is used in conjunction with fragmented packets, this field indicates whereabouts in the packet is the actual data stored. This is measured in octets.

8 Time To live - This field is set by the computer and is decremented as it passes through a router. If this is set to zero, the packet is discarded. Generally speaking,

routers should never receive a packet that has this field set to zero otherwise a network loop will occur.

9 Protocol - This indicates which transport layer to use, six is the more common value for TCP.

10 Header Checksum - This contains a value to ensure reliability of the packet. Note however, this excludes the data, there is a reason behind this, and the checksum must be recalculated as it passes through the router which can slow transmission time.

11 Source IP Address - represented as in dotted quad hexadecimal notation, e.g. AC 10 C8 FA_{16}, in base ten, this is represented as 172.16.200.250. In binary this would be$10101100\ 00001000\ 11001000\ 11111010_2$ or simply www.google.com which is the more familiar way of representing computers on the Internet.

12 Destination IP Address - again, it is in the same format as above. See number 11.

13 Options - this is useful for examining the packet, there are a number of options available concerning measurement, security and debugging.

14 Padding - this is to ensure that this section of the packet is evenly 32 bits long.

15 Data - the actual data, the length is variable as long as it does not exceed the limit.

2.2. Introduction to Client/Server Models.

Client/Server is a means of allowing clients access the server to use resources. The client, can be classified in terms of hardware, software, or both, and is usually interactive with a user. In addition, it can reside somewhere on a LAN. The servers on the other hand, sit somewhere, again on a LAN or even on WAN, and require little or zero interaction with the user. The best definition of Client/Server is “a client initiates an action with a server by sending a message, it is independent, may run on the same machine, the hiding of information inside a message. The server enforces certain rules on how data is arranged so that the client can interact with it", ***(Umar, 1993, pg 247)***.

How data is enforced is called marshalling/unmarshalling. This is how it works, marshalling is about putting data together, which is what the client does, and it sends the message containing the data to the server. The server checks if the message and data conforms to certain rules unmarshalls the message, i.e. Breaks it up, into a certain way for processing.

Here is the outline of Client/Server's advantages and disadvantages.

Advantages.

- Client/Server efficiently divides functions among diverse platforms - the server does the majority of the hard work in processing high-volume data providing integrity of the data, therefore, the client becomes a more effective, productive machine without wasting

    a) Valuable CPU cycles in processing
    b) Database storage is virtually nil or kept to a minimum

- The use of database processing and transactions empowers the client to interact with multiple databases on multiple platforms; thus, application portability is retained.
- Servers are scalable and can be easily customised without breaking the client, i.e. The client does not need to know what is the server's specifications, as long as "there is a server listening to me", and is configurable. The client does not need to be upgraded at all.
- There is a huge saving in terms of cost in transaction processing. Mainframes tend to have higher cost in this regard, e.g. price per transaction on a client/server is ½th in comparison to 1/10th of that on the mainframe.

Disadvantages.

- Because of the specific nature of client/server systems, complexity increases in terms of network management and operations; thus, this would require recruiting skilled specialists to support it.
- Application packages and development software is scarce and expensive. Thus, the cost of software development is high. Different software vendors have their own standard (which brings about the Open Software Foundation - OSF).

Generally, software that uses the client/server model is custom/ vertical/specialist software that would not fit in the category of buying it off the shelf.

There are different models of how a client/server system would work. From a software development point of view, each of these models requires a certain level of technicality and possibly having to retrain the knowledge from time to time.

2.2.1. Pipes.

Pipes are strictly speaking, from a Unix based point of view, is a special file, in which can be accessed for reading/writing to. This is the simple way of communicating. A server designates a file, opens it up for reading/writing, and waits until data is coming through. The client opens the same file for reading/writing, writes data to the file, once the data arrives on the server; it reads from it and does whatever is necessary. There are advantages to this

- No special skills are required, if you know how to read/write to a file, then you are done.
- It does not use TCP/IP protocol.
- Simpler and cheaper to use.
- Different flavours of Unix (such as AT&T, System V, AIX, Linux, to name but a few) already have special facilities to create this special file using their programs.

2.2.2. Sockets.

Sockets was an extension to pipes but is more generalised to communicate across a network, be it LAN or WAN. A socket application simply reads/writes to a socket to communicate with a server; this is akin to performing read/writes to an ordinary file. There are a number of well-known applications that uses sockets, telnet, and ftp to name but a few. The socket implementation is the more natural approach for developing multiple clients communicating with a single server. There is an already established standard of sockets, which conforms to the Berkeley standard. Windows platforms use this standard via Winsock, which also includes proprietary functions to

take advantage of Windows. This means, that if you write an application that uses Berkeley style sockets, the code will be portable across the Windows platform.

Advantages.

- Portable and clean interface, and is well known due to adherence to Berkeley standards.
- Can accommodate TCP/IP and UDP protocols.
- Tighter control over how communication protocols work.
- Majority of systems comes with means of creating socket applications.

Disadvantages.

- Requires a level of technicality, and intimate knowledge of the network protocols.
- Developing a socket application can be complex, such as threading, handshaking, and even arranging how data is to be transferred.

2.2.3. Remote Procedure Calls (RPC).

RPC, in a client/server model, looks like a local procedure routine. "RPC function calls are function calls outside of you address space that look and feel like local procedure calls", ***(Bloomer, 1995).***

When that routine is executed, what happens really is the underlying code or mechanism performs communication with a server across the network. This adds a level of transparency, to which the programmer need not be concerned with the technical details of the network infrastructure.

The server will contain the routine that shares an identical name to the one on the client, this is where the server will do the necessary work in processing the data, and the results are returned back to the client.

Advantages.

- Information regarding the routine/function contains the network mechanism is automatically generated by a RPC code generator - similar to that of a code compiler.
- Majority of commercial software suppliers follow the OSF/DCE standard thus it is portable.

Disadvantages.

- Extra work involved in having to specify an interface definition, which is then passed into the RPC code generator.
- The development lifecycle can get complex regarding debugging/testing the application. "Distributed application debugging can be very challenging…It's extremely productive to first link the service procedures directly with the client side of the application….debug parameter passing and overall functionality can be exposed by side-stepping the network and RPC calls…" ***(Bloomer, 1995).***
- The software can be expensive to set up in terms of security, authentication etc. Code can be complex especially in a security conscious environment, with authentication hard to verify.

Chapter 3.

3.1. Introduction to Case Study

This chapter is focused on a company located in the Mid-West where I spent almost five years working in a number of areas, specifically Client/Server development. The objective of this chapter is to give a few factors such as training, ease of use, problems, mindset of management that is outlined in 3.4 and 3.5 below.

- A background on what the company does and what role it plays
- A brief outline of how American Healthcare system works.
- IT Infrastructure where relevant.
- Problems if any & solutions.
- Other miscellany issues, which I feel, are relevant.

3.1.1. A background on what the company does and what role it plays.

This Irish company works in tandem with the central office based in Hartford, CT, U.S.A, essentially an R.P.F - Remote Programming Facility, where tasks are shared in conjunction with the U.S, working as a team.

The central office plays a big role in the insurance healthcare sector producing and maintaining software to support the existing system. As regards to this sector, this is a very big and complex operation.

3.1.2. A brief outline of how American Healthcare system works.

The complexity lies in how the health insurance is set up due to differing laws as agreed by either U.S state law and or by U.S Congress/Senate. Of course, this can change every year depending on the approval by members serving in the health sector, such as doctors, hospitals, pharmacies etc. I will give an outline of how health insurance works in the eyes of the United States.

Typically, all employers will offer their employees some form of incentive or benefit, in this instance, health insurance, typically under a number of different packages depending on the company itself and the number of employees. There are many packages such as HMO, PPA, PPO, MIN, to name but a few. Please see the list of glossary terms for more information.

Each of these packages would have what is known as a 'Plan', and each 'Plan' would have certain 'Options' (if any depending on employer/employee contribution).

A 'Plan' is made up of and agreed by a 'Network' (nothing to do with I.T!). A 'Network' consists of 'PCPs' (Primary Care Physician), Hospitals, Pharmacies, Nursing homes etc., and is demographic based. See Figure 3.1.

E.G.

Health Package offered by Company Z, located in California, U.S.A. Employer contribution per month - $100, Employee contribution per month - $75. This health insurance shall be called HMO Plan with Options MHSA (Mental Health/Substance Abuse), Neo-Natal Pre & Post Care.

It is worth mentioning that on the one end of the spectrum, there is HMO that is considered to be the best health insurance an employee can get (depending on their salary which would be high obviously). On the other end there is MIN (Managed Indemnity - this is where all employees pool their contributions together) for low-paid employees. Incidentally, under the HMO Plan, PCP's, hospitals and such often get malpractice suits and have a bad rap. This is very much the opposite case for MIN Plans!

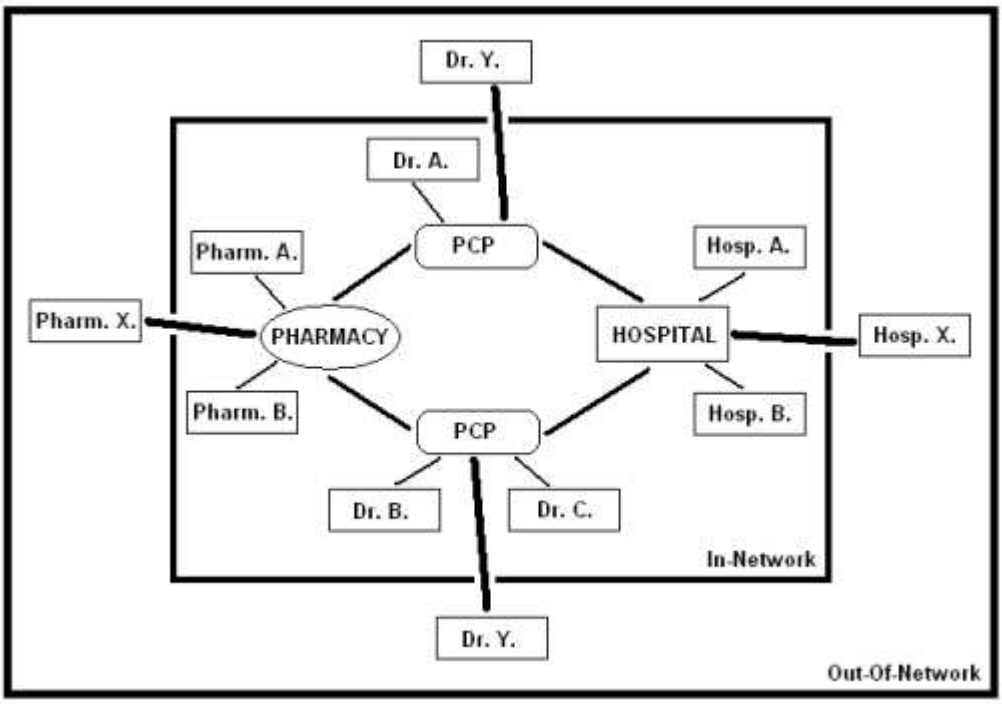

Fig 3.1. Illustration of a Provider Network.

To further make the Insurance package more complex, an employee who works for Company Z in California, has HMO Plan (remember it is demographics based). And that employee's health insurance would be valid and effective within California or in health insurance speak - within the 'Network' or 'In-Network' (cheaper and the insurance company would cover it). But it may not be effective outside of California, for instance, New York. Even though a HMO Plan in New York is deemed valid, but since it is outside of the 'Network', such coverage in emergency (for instance, if the employee breaks a bone or worse whilst visiting his/her relatives in New York) is not possible. Therefore it will be more expensive and the insurance company may not or refuse to provide such cover. Hence it will have to come out of the employee's pocket!

As you can imagine, 'PCPs', Hospitals, Pharmacies would have to negotiate with the insurance company to provide 'Network' coverage and agree on fees etc. Furthermore, employees who have HMO Plan as in the above example must visit their 'PCP' while 'In-Network'. Moreover, the individual must attend a certain hospital per se etc., in order not to pay - i.e. insurance company will look after the bills.

Yes, 'Out-Of-Network' coverage is also available but will be charged slightly more. For instance, an employee who is within three miles of 'In-Network' coverage will pay nothing, in case of emergency and is twenty miles outside of 'In-Network' i.e. 'Out-Of-Network', the necessary fees will have to come out of their own pocket.

That is the brief coverage of what the company is the background of its role within the I.T/HealthCare sector. Now we will look at its IT infrastructure pertaining to the area of Client/Server technologies, it is problems and other issues.

3.2. I.T Infrastructure used in this Case Study.

The platform is IBM's RS/6000, running AIX 4.3 and DB2 Universal Database, the backbone of the LAN/WAN is fibre optic via Ethernet hub/switch with T1 link (Tie line which has huge bandwidth). This enables connection between the Irish Office and Connecticut, U.S.A. The RS/6000, now have been renamed to the pSeries which was part of IBM's marketing move, mainframes such as IBM S390/AS400 are now called the zSeries. For a sample of such system, it can be found at IBM's website.[1]

There were two RS/6000 boxes, which is located off-site for security reasons. The production box serves as the access point for different offices located in Colorado, Texas, Minneapolis, California, South Carolina, and New York. It has a thirty-two-way processor running in tandem, with twenty-four gigabyte of RAM and storage ran into a couple of terabytes (i.e. a very big number).

As you can imagine the sheer processing power, required to handle requests from different states within demographic regions are massive, in particularly, there's a very high database transaction processing, which explains why it has a very high specification.

The specification of the development box is slightly lower than that of the production box, and there is a reason why, for performance issues mainly, i.e. if the system can

[1] Entry level RS/6000, available on internet, last accessed 13/11/2002, *http://www-132.ibm.com/content/home/store_IBMPublicUSA/en_USA/eServer/pSeries/entry/pSeries_entry/6106E1*

work under this specification. It can be deduced that it will perform better on the high end. It has sixteen processors with sixteen-gigabyte RAM and storage space was about the same. IBM has on-site support ensuring both boxes have minimum downtime per year and upgrading them where applicable.

The development box is where all the change control procedures take place that involves modifying, testing, and debugging the applications. Borland's Entera for AIX runs on both boxes, all server code is written using 'C', database access uses SQL and the server code, and database access resides on the one box (both in production and in development). The client is based on Powerbuilder 6 running on Windows NT platform, which is albeit a very complex application (as I have mentioned above in describing how the health system works, the complexity of the user interface is enormous). However, it works nicely; the user interface takes a bit of getting used to nonetheless. Entera is used to communicate between the client and server, which will be part of this discussion for this dissertation.

3.3. Development lifecycle.

This will provide an insight into how the development lifecycle works, in relation to using the commercial DCE software. Entera encapsulates the network specific routines via client/server stubs that are generated at compile time. In this case, the developer is more focussed in getting the logic coded thus freeing up any worries/headaches about networking details. This was all down to the way the compile process was done.

A brief explanation in how this was achieved. All RPC functions were known as a business functions (which was written in 'C'), this in turn call other routines to manipulate the data passed in from the client side, i.e. Powerbuilder and the database. The business functions were added into an interface definition language file (IDL), and a code generator, creating client and server stubs, processes this file. Also generated was the Powerbuilder import file, which is transferred across to the PC platform for inclusion during the build process of the Powerbuilder front end.

Once, the server stub is compiled into an object code and linked in with other libraries, a server executable is created. The server executable listens in on a specified port number running in the background, blocking or waiting for a connection from the client side. Therefore, the client side must be told which port number to use on the host platform on the network. This in turn, this will execute the business function on the server side depending on a certain event in the front end.

Entera already has a graphical user interface debugger specifically for debugging RPC functions known as `rpcdebug`. Of course, the debugger must be told which port to listen in on, and the developer loads up the executable into the ordinary debugger that comes supplied with Unix. By setting a particular breakpoint and running the server in tandem with `rpcdebug`. The developers can specify which RPC function to execute (the IDL file must be specified as well so it will know which functions are relevant to the particular server). And supplying parameters and simply click on the 'Execute' button, the normal debugger intercepts the RPC function in question and from there, debugging is relatively easy, single stepping into the code (white box testing) and function stepping (black box testing).

There is another way to debug a server without using `rpcdebug`, this can cause the client side to lock up completely and hang which would result in the need to kill the process. This stage is usually carried out, once the developer is satisfied that the code executes using test data. The business function would have to be altered slightly, simply by including an infinite loop like as shown in Table 3.1:

```
int i = 1;

..... Certain start-up house keeping functions....
while (i); /* Breakpoint set here in debugger */
..... further code....
..... and certain shutdown housekeeping functions....
```

Table 3-1. How an infinite loop aids debugging.

Go through the stages of compiling/linking to rebuild the server. Load the server executable under the debugger and setting a breakpoint on the line that contains the infinite loop (`while` loop) and let it run in the background. On the client side, create

the event that will cause the business function to be executed on the server side. The debugger will break at the specified breakpoint, and by setting the value of `i` to zero, one can resume debugging in the normal fashion, checking the parameters etc.

This is the reason why there is an infinite loop, business functions gets executed very quickly and thereby guaranteed that the code can be stepped through once the value of `i` is set to zero. This explains why the client side can be locked up completely and hangs as if the client has crashed!

3.4. Problems and Solutions.

Entera 3.2 was used at the time, and Inprise (now known as Borland) dropped the support for this version. Furthermore, project managers perceived a risk in upgrading Entera 3.2 to Entera 4. This was considered unjustifiable. At the time, it was not justifiable in investing money, as it was too expensive to continue using the software that would get no support and was seen as a major risk.

For instance, what would have happened, if a broker/cell went down due a software malfunction that was seen for the first time and occurred at random thereafter in which support has been dropped? This is something that management perceived, and yet would not be willing to lose money in terms of services impacted by ‘down-time’.

Open Source was never heard of at the time and thus was not seen as alternative. The solution at the time was to ditch Entera, and use Java/Javabeans/HTML. It was considered a huge investment, as many people in the project had to switch over to new grounds. Thus, the development team started to get smaller, and receive cross-training classes on Java and to migrate many of the core routines over to Java via code wrappers. Short term, it was expensive (cost of training, team brainstorming discussions such as how to minimise re-inventing the wheel with ‘C’ and Java etc). Long term, it was seen as a good alternative and was decided to go down that route.

3.5. Other miscellany issues, which I feel, are relevant.

Training was minimal due to complex information hiding regarding DCE, as to what was involved was learning how to use their make-files, understanding command line parameters when starting up/shutting down servers.

Overall, the development lifecycle was easy to do. There was no complexity involved as the system administrators off-site were maintaining the broker/cells. As well as one or two privileged members of the development team who shared a subset of responsibility in the event that the system administrator was absent.

The compilation of code was minimized due to intelligent macros in make-files, which were set up. The developers were not worried about losing productivity; in fact, there was no productivity lost due to the intelligence of the software and the ease of use. Developers (including myself) did not see the lower-level RPC routines, as this was all hidden away inside the stubs, which were generated at ease.

Testing and debugging the code was easy in general; again, developers need not worry about the underlying mechanisms regarding Entera. It would be fair to say that the edit-compile-link-test-debug cycle was smooth and no extra knowledge is required in how the underlying principle works. Ok, there was a few commands that has to be remembered in setting up a sub-broker (See next chapter for in-depth explanation), and is the same across for all types of servers.

Chapter 4.

This chapter will focus on the different commercial products; I will give details how the underlying architecture works based on my work experience. For now, I will give a brief outline of Entera and Entegrity by gleaning in on the reference manuals (in Acrobat's PDF format). Which I downloaded during my research and then I will go into more depth about Entera. I will focus on two products, one for the commercial Unix such as AIX and the other for the Open Source known as Linux.

Some of the factors will be highlighted here at the end of this chapter, for commercial DCE software such as:

- Cost Issues
- Ease of Use
- Support/Stability

4.1. Entera

What is Entera?

Entera is a middleware product that developers use to create three tier client/server applications. Entera conforms to the Open Group's DCE standards, which makes it a portable and interoperable product. In other words, code developed using Entera should work with any other Open Group compliant DCE standard. Strictly speaking, in this case I mean the source code can be built under a different DCE architecture provided it conforms to Open Group standards. Its tools insulate the developers from having to know the underlying network protocols and infrastructure; hence, Entera applications are independent of the network infrastructure. ***(Borland, 2002).***

Entera & Databases.

Entera can interact with many different databases such as Oracle, DB2, Interbase, to name but a few. Thanks to its data access servers, it can access Open Database Connectivity (ODBC) compliant databases as well.

Entera provides support for:

- Automated transaction handling.
- Stored procedures support.
- Automatic SQL statement caching to improve performance.

Entera supports different languages.

Entera enables developers to develop and deploy three tier scalable applications under many such diverse languages such as:

- C, C++ (Borland C++ Builder).
- Java.
- Visual Basic.
- Powerbuilder.

The language support is due to its dynamic link libraries (DLL's) and Add-ons.

Entera is scalable and secure.

Entera provides for DCE Delegation support, which allows security credentials is propagated across security servers enabling services such as single sign-on security. It offers flexibility, and scalable, supports asynchronous RPC's which frees up the application to process something else while the RPC is executing in the background.

4.1.1. How Entera works.

In Figure 4.1, the illustration is shown how Entera can interact with the client. On the left of the illustration, is an IBM RS/6000 box running AIX with the main broker(s) or cell(s). In this example as shown, the main broker is running off TCP port 9876. Inside the main broker, there can be one or more servers – up to **n** servers written in 'C' with Entera server stubs. All of the servers provide a means of interaction depending on what event was triggered on the client. For example, clicking on a drop down box, this in turn invokes a certain routine on the server designed for handling drop down boxes.

On the right, is an Intel x86 running Windows NT 4 WorkStation, with the client running. The client is written using PowerBuilder with Entera client stubs and is specified to use port 9876 via an information file, similar to a common file that is found under Windows platform called `win.ini`.

The client stubs were generated on the server and exported across to the x86 platform. This 'client stub' is not technically a stub in case you are wondering how it is achieved! It is more of an ASCII flat file with routines to hook up with the Entera DLL residing on the client.

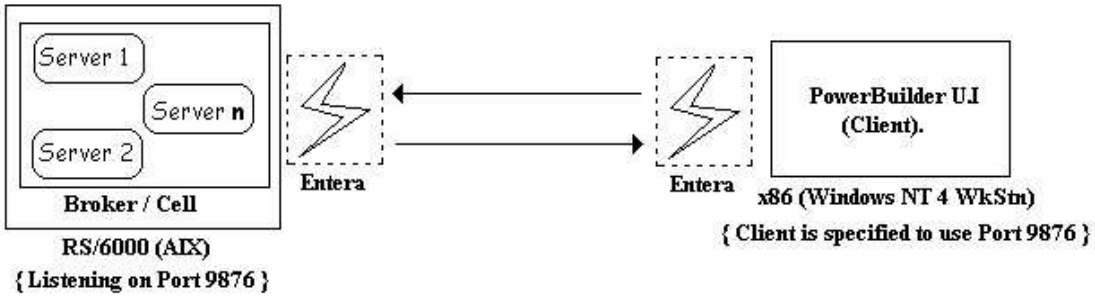

Fig. 4.1. Illustration of Entera.

A very good example of this will be shown in a different language, the principle is similar. Let us look at Microsoft Visual Basic, to give you an idea how this is done. Here is a brief introduction about programming in VB with API. You want to use a certain API (Applications Programming Interface); to change the behaviour of a certain event. Alternatively, to use a certain feature not found in VB, one could code it

in this way. Notice in Figure 4.2, how the library user32 is referenced which meant, declare a function called "`FindWindowA`" which can be found in "`user32.dll`".

When an Entera client stub is generated, it uses the similar principle; it just contains references to the Entera DLL, along with the RPC/business function names. The stub is then imported within PowerBuilder. Obviously, the references to the Entera DLL would remain the same, and thus would be built into a separate library and hence would be discarded.

```
Declare Function FindWindow Lib "user32" Alias "FindWindowA" _
(ByVal lpClassName As String, ByVal lpWindowName As String) As Long
```

Table 4-1. How Visual Basic taps into the API.

4.1.2. How a main broker can be sub-brokered.

In this scenario as shown in Figure 4.3, a programmer wishes to modify server 1. He/she then sub-broker it to run off port 8765. The assumption with this scenario is that there is no change made to the RPC/business function. Just the underlying logic is modified. The client (since no changes are made) is specified to use the programmer's port 8765.

This is where things get clever. Suppose that a user on the client triggers a certain event i.e. calls a RPC/business function that is not running off port 8765. The RPC/business function is found in "Server 2". Entera then re-routes the request for the RPC/business function to port 9876, the code residing in "Server 2", are executed, returns the results back to the sub-broker running off port 8765. The client is not aware of this and thinks that "all" servers are running off port 8765 whereas in fact only one server is running! Furthermore, other clients can still use port 9876.

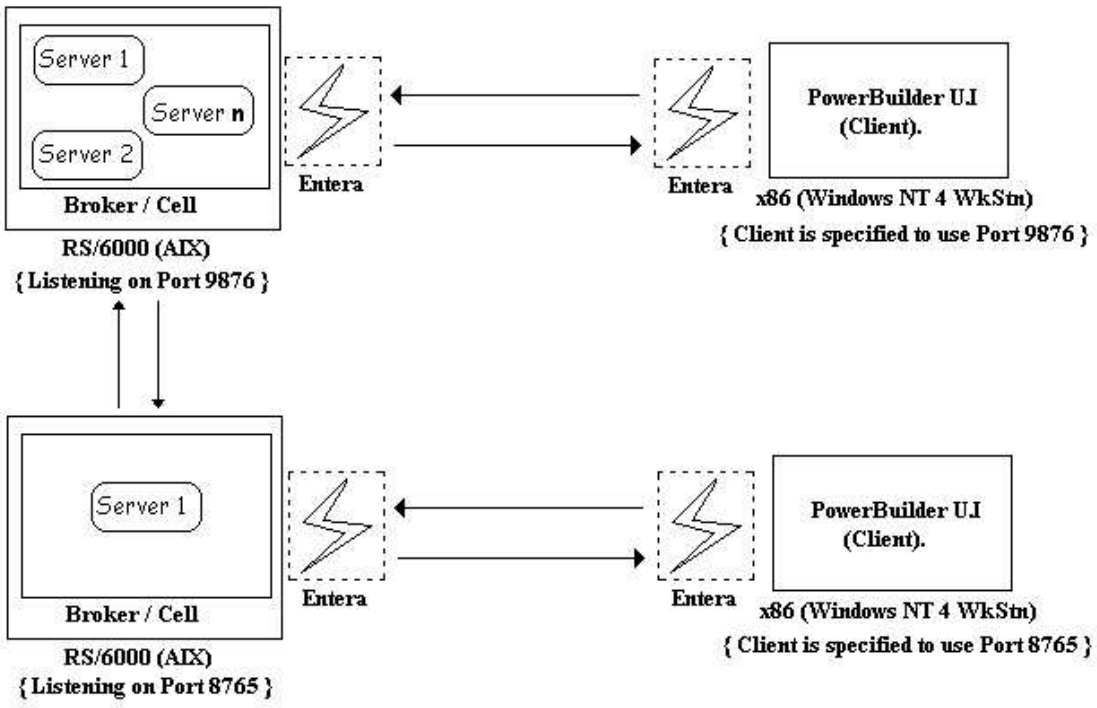

Fig 4.3. How Entera can be sub-brokered.

4.2. Entegrity.[2]

Entegrity is by far, the only commercial DCE product available for the Linux platform. The crucial difference is that, it is client development toolkit minus the server features, i.e. there is no means to set up a cell or broker, ***(Entegrity, 2002).*** In other words, you can develop DCE client enabled applications, and integrate it with an existing DCE software system in place. As of now in 2003, I have been told that there will be a server version available.

Entegrity is actually based on the OSF/DCE version 1.2.2 source code ***(OSF/DCE, 2002)***, in which FreeDCE is a descendant of ***(FreeDCE, 2002)***. The binaries for this software is available for a number of operating systems and Linux distributions, such as Windows, SuSE Linux 7.3, RedHat 7.2 to name but a few.

There is an evaluation kit available for download on Entegrity's web site; the evaluation only consists of a number of header files for inclusion with software

[2] Linux DCE Client, available on internet, *http://www.entegrity.com*

developed in 'C', and an interface generator. The kit expires after thirty days after downloading.

When I downloaded the software, I was expecting much more, since the software requires an existing DCE cell, there was no way I could use it to build a simple client/server application. Once the server version is available, I will endeavour to try it out.

This product has yet to reach its potential. In my opinion, once it has a server development kit included, then this could open a new market, in an area of maintaining client/server applications under legacy systems. Thereby bringing it across to Linux to give it a new lease of life, there is room for growth here, which has yet to be seen.

I will highlight briefly the factors I feel is relevant and pertaining to the case study and to this chapter.

Factors:

- *Cost Issues:*

  For a software development team that comprises of fifty developers, the cost will increase as regards to the number of licences. Licences are varied concerning Entera, such as how many servers will be running (this does not apply to Entegrity yet). How many clients are using the DCE middleware (In the case of the case study as highlighted in the previous chapter, we're talking about a number that cannot be determined so I'll call this variable as NoClients).

  Licence cost for developers = 50 x Single_Developers_Cost_Of_Licence ***(A)***
  Licence cost for end-users = NoClients x Singe_Client_Cost_Of_Licence ***(B)***

  Also, the cost of support must be considered here, as in the case study, on-site support offered by IBM Global Services excluding the cost of supporting the boxes alone, and on top of that, yearly support by Borland, which can be

perceived as a huge investment. The TCO (Total Cost Ownership) applies but is not effective enough to warrant the measure of overall cost. This will be explained more in Chapter 7.

On-site support of Boxes = Z x Cost per Year ***(Z)***
On-site software support = Y x Cost per Year ***(Y)***
Technical Support by Borland = S x Cost Per Year ***(S)***

Total: (A) + (B) + (Z) + (Y) + (S)

Note: A and B would be considered to be once-off, as usually, Borland would offer a discount for support for say three/five years (depending on how long they would support the software for)

Any additional licences such as expansion of development team is not taken into consideration here, but as you can see in the simple maths above, this can work out to be quite expensive on a yearly basis!

- *Ease of Use:*
  The biggest factor would be the role of the System Administrator responsible for setting up the software, including DCE administration. While they make things easy for the development team, the extra level of work involved would be removed.
- *Support/Stability:*
  Again, the role of system administrator would come into play here, maintaining DCE software, applying patches as well. Inherently, it would be cheaper to hire a dedicated Systems Administrator long-term, rather than getting on-site support which can prove costly, based on a per-yearly basis.

I will go into more detail in covering the factors that will make up the overall conclusions that will be in Chapter 7.

Chapter 5.

This chapter shall focus on the Open Source movement, who founded it and give an introduction into a very well known, if not, famous project that currently exists today. In Chapter 3, I have looked at the commercial software Entera and Entegrity, now I will give also give an outline of the Open Source alternative projects in relation to Client/Server Development technologies.

Some of the factors will be highlighted here at the end of this chapter, section 5.6, for Open Source DCE software such as:

- Cost Issues
- Ease of Use
- Support/Stability

5.1. Introduction to Open Source.

Open Source software is a loosely based term used to refer to Free Software. Free in the sense that end-users can make multiple copies of the software, free to copy it and distribute it, to have the freedom to choose which software without getting tied in or locked into proprietary software. The majority of the Free Software today comes with a copy left licence, more commonly known as GPL. ***(GPL, 1991)*** GPL is an acronym for GNU General Public Licence, which was published in 1991, GNU is an acronym coined by a well-known founder of the Free Software Foundation ***(FSF, 2002)*** - Richard Stallman - See below for further information.

GNU is pronounced as "Guh-nooh", it started as a humorous means in relation to a technique in programming called recursion. It is only that computer literate people could understand. A recursive acronym simply states **G**NU is **N**ot **U**nix. ***(FSF, 2002)***

5.2. Confusion between Open Source and Free Software.

Currently, there is still widespread confusion over Free Software and Open Source; individuals are still debating whether they mean the same thing from time to time thus creating a massive argument, especially in mailing lists and Internet chat rooms. Although they may look similar, and it would be safe to use both meanings interchangeably provided if the software:

- Is free.
- Comes with source code as provided
- Can be modified and distributed without any restrictions.

From now on, I will use the term Open Source to denote software that is free and code is available bundled with the software.

Here I'll outline the major crucial difference between Open Source and Free Software, Open Source software comes with source code for individuals to see, but may come with a license that forbids to distribute it or modify it. Whereas Free Software comes with source code, but also comes with GPL which states "having the freedom to use, modify the software, distribute the software at a profit including source code" (See Appendix A for GPL in its entirety). I do admit it can be a quite complex area and avoid getting into a discussion about it with others, as individuals will have their own meaning and interpretation of Open Source and Free Software.

The licence themselves enclosed along with software (whether commercial or free/open source), can only be read and understood by Lawyers and not generally members of the public. To quote from the GNU Bulletin, “The Free Software Foundation is dedicated to eliminating restrictions on people’s right to use, copy, modify and redistribute computer programs”. ***(FSF, 2002)***

The majority of free software that comes with it’s source code will have a licence similar to GPL, with an exception, if a software company wants to release source code for their proprietary software i.e. Open Source it; there will be restrictions or

exclusions from the GPL such as not having the right to redistribute it. There can be variants of the licence such as the one supplied with FreeBSD[3] (another Unix clone complete with source code, similar to Linux and comes with Berkeley licence), but not affiliated with FSF. As it is a very grey area, a lot of individuals would rather choose software that follows the spirit of FSF. I have enclosed a copy of the GPL that can be found in the Appendix section, for perusal and interest. It is worth mentioning, that the code I have used as part of the implementation for this Dissertation is under the GPL and is free to modify, share, copy, distribute, the only onus is that my name gets a mention!

Software that is Open Source tends to be better built, robust and with fewer bugs. I will quote a fact or two as per on the paper found on the Internet. ***(Mockus, 2002)***.

- Open Source projects tend to have a core team, which is based on three-tier layer, the bottom layer consisting of ten to fifteen coders producing almost all of the code. The next layer above that is a set of developers submitting new features or bugs fixes. The last layer is a set of advanced users submitting bug reports and testing the software. (See Figure 5-1, An upside down triangle)
- Open Source projects tends to have a lower rate of bugs than that found in commercial software due to code being open to public scrutiny, "Given enough eyes, all bugs are shallow" ***(Di-Bona, 1999, Cubranic, 2002)***.
- Open Source projects are generally quicker to respond to user requests, applying bug fixes.

Fig. 5-1. Illustration of hierarchy of Open Source development.

[3] FreeBSD can be found at *http://www.freebsd.org*

5.2.1. About the founder of GNU, Free Software Foundation.

Richard Stallman, is the founder of the GNU /Free Software Foundation (FSF for short), and it started as a project in 1984. He was a very creative but politically motivated individual who was considered a gifted programmer at his time. GNU, FSF was established after Richard Stallman resigned from the Artificial Intelligence department in Massachusetts Institute of Technology (M.I.T) for twelve/thirteen years, maintaining an unknown operating system at the time called ITS (Incompatible Time-Sharing System). ***(Stallman, 2002)***.

Shortly after FSF was set-up, he wrote a series of GNU software specifically, which was the first of its kind to come out, the portable, extensive compiler collection that consisted of optimising compiler, a debugger, and an editor. (The GNU C compiler known as gcc, the GNU debugger known as gdb, and GNU Emacs respectively).

Today, the gcc software can produce code for a host of diverse architectures and different programming languages, currently, thirty architectures and seven programming languages so far[4]! In addition, gcc is the primary must-have software when it comes to developing a wide range of software, such as Linux. Without gcc, Linux or any Open Source Software would not have got very far let alone be heard of.

5.3. Is Business ready to take on Open Source?

At present, there are individuals that are very knowledgeable in the fields of technical and business nature, recognise that Open Source is gaining immense popularity due to a number of factors. Greater range of software, that is of better quality than their commercial counterpart. It has lower costs, in terms of licences and immediate responses to bugs, which create better and robust software unlike in comparison to proprietary, closed software companies. Proprietary companies are known for delays in acknowledging that there are bugs in their software. Issuing fixes leaving things a bit too late, while the end-users simply workaround the bug(s) which can have detrimental effects in terms of productivity and frustration. ***(Kozinski, 2002)***

[4] http://www.fsf.org/software/gcc/gcc.html

The primary drive behind the decisions to take on Open Source in some companies is the cost factor. For instance, there are many software packages that are considered robust and perform better to the commercial flavour. Look at Linux, which we will be looking at shortly, is an operating system that has no licence costs. And it can be rolled out on to hundreds if not thousands of computers without living in fear that BSA will be knocking on the door demanding to know if such software are legitimate and have valid licences. Linux can be copied without fear of persecution.

Look at Apache[5], which is free web server software, many people tends to forget that Apache account for nearly 80% of web servers on the internet. Alternatively, what about GIMP[6] that is an image-editing software. In terms of features, it is a rival to Adobe PhotoShop. On the other hand, even, not to mention Open Office[7] that is an open source version of Sun's Star Office suite that is commercial. Open Office, is also a rival to Microsoft's Office suite, (and even better, it can load/save many different file formats). Think of the potential savings, reaped in terms of cost of licences, which will be outlined in Chapter 6, e.g. Microsoft's Office Professional Edition costs 850 Euro. Now, suppose that there are 200 computers running Windows 2000 and Office needs to be rolled out, that's 170,000 Euro for licences alone and for keeping BSA happy! With Open Office, no licence costs, but the trade-off in terms of training would have to be considered - not bad!

As for quality & robustness of such Open Source software, one can be very forgiving at the pace of releasing bug fixes in such short space of time. Thanks to the nature of the Open Source model and from a software development viewpoint. Many programmers will certainly hold the source code to such software under scrutiny and thus bugs would be eliminated resulting in stable, robust software. Furthermore, the code is highly unlikely to be a virus, (that at once stage was a myth back three/four years ago!). Hence, the code itself is guaranteed to be free of sneaky, devious code that shares similar qualities of viruses.

[5] Apache's web site, *http://www.apache.org*
[6] The GIMP's web site, *http://www.gimp.org*
[7] Open Office's web site, *http://www.openoffice.org*

Business needs to be aware of two things that can result in disaster if they try to integrate Open Source software into the business. Failure to take into account of them can result in disaster. The first is to ensure that the continuity of the business operation is not affected. For instance, having to schedule downtime, which can be costly and the second thing is to ensure that there is a training program in place, usually in parallel to the integration of the software.

Of course, businesses would have to be aware that there might be resistance to change. For instance, integrating Open Office, a commonly held view is; "I am very fond of Microsoft Office and fear of losing productivity if I switch over to this Open Office thing". This will have to be dealt with via education. Start in small steps taking one step at a time to ensure success, continually educating everyone in the process, getting everyone involved in the integration of such software. In that way, success can be guaranteed in the end where costs will gradually diminish in terms of licence costs. Furthermore, with such success in time, grievances concerning piracy issues will be outdated, and even for smaller companies who cannot afford the licence costs usually end up resorting to using illegally copied software. ***(Drummond, 2002)***

5.4. Introduction to Linux.

This has to be the most famous known project today. It is part of the GNU, Free Software heritage. Linux started in 1991, when a Finnish student by the name of Linus Torvalds was studying computers in Helsinki. He was playing about with a variant of Unix called Minix (written by Andrew Tanenbaum, to teach how operating systems are written) that was designed for the PC.

Apparently, and this will lead to certain amount of debate, Linus was not too happy with Minix's scheduler. He modified the code heavily and Linux was born. There are variants of this story of how it all got started. Linus claimed that he wrote the system from scratch in order to learn about the architecture that was Intel 386 at the time; however this debate would not distract me.

The first kernel was devised around April 1991 as a hobby initially. After a short period after making it work, version 0.01 was available and functioning around

September 1991, despite lack of features that would make it resemble Unix. It was until circa January 1992, after coding, debugging, that kernel 0.12 was released and had a lot of Unix-ness to it. Then finally after a lot of fixing/patching up code, kernel version 0.96 was released circa April 1992 when Linus posted an email to a newsgroup. The code was released under GPL, it received so much attention that developers all over the world clambered on board to help out and shortly after that, it hasn't been the same ever since. *The timeline is not entirely accurate; please see the Appendix D for more detailed information from early newsgroup postings announcing Linux.*

As of today, there are approximately 18 million users of Linux, and the most current kernel release is 2.5.30 (this is state of the art/bleeding edge). Stable wise, kernel release 2.4.19 is by far the popular release. And yet as ever, there are more individuals joining the ranks of developers around the world, in helping to maintain the kernel so Linux will be around for a very long time.

GNU/Linux or Linux, in short, is an operating system that aims to comply with POSIX standards and shares many features that look and feel like Unix. The source code is freely available so that technical minded individuals would enjoy and learn from it in how it works. It comes with many features such as different file systems support, integrated network protocols, wide range of assorted hardware support, which would be very extensive and beyond the scope of this dissertation.

Linux also runs on various platforms, namely Intel x86 family processors, Motorola, Advanced Risc machines, Apple, IBM S/390, not to mention it can also run on Microsoft's latest console box - Xbox with a bit hacking! It would be fairer to say that Linux in itself is the kernel. The utilities and programs that come with it are what makes up an operating system is called a distribution. Hence, when individuals talk about security exploits in Linux, they are actually referring to the utilities (which do have security problems - commonly buffer overflows), not the kernel itself. There are many distributions available such as SuSE, RedHat, Slackware, to name but a few.

When individuals discuss about Linux, a reference is made to either a distribution or the kernel depending on their topic of discussion. The kernel is the essential core of

the operating system or the heart of controlling the computer. There are two types of hearts in a computer. One is hardware called the microprocessor. The other one is the software - the kernel that has a responsibility in communicating with other hardware devices, keeping applications running, as such. Linux is used in many different areas, such as network servers, embedded systems, software developments, and database and application servers. Linux is also used for designing vertical solutions such as hotels, hospitals, and governments, ISPs etc.

In keeping with the traditional spirit of Open Source, Linux is the base platform of choice for implementing a client/server project and for writing this dissertation.

5.5. Open Source DCE.

There are two variants of DCE software technologies, OSF/DCE, and FreeDCE. The major difference between the two is in terms of licensing; OSF/DCE is used only for academic and research only, whereas FreeDCE is under GPL. The source code is available in both versions. However, take notice of the inner details of OSF/DCE's licence – note academic and research only! I will mention general aspects of the two of them in this regard.

5.5.1. OSF/DCE.

OSF is a non-profit organisation and is made up of a consortium of Unix vendors such as IBM, DEC, and Hewlett-Packard. Its primary goal was to come up with a means of standards for Unix; it was realised standardising Unix was not enough to standardise distributed applications. I felt it was important to mention about this, since the base code of OSF/DCE is shared between Entegrity and sourceforge's FreeDCE.

OSF changed tactics and instead, focussed on interoperability between multiple products from multiple vendors. As such, different vendors have their own standards that would make software development a nightmare. In terms of porting code across from one platform to another in the sense of having to change code to reflect the vendor's own set of standards. "Middle-ware like DCE lets users get away from the multiple vendor, multi-product approach to distributed system integration. As more

vendors build DCE into products, integration and distributed, system development becomes a non-issue. 'No doubt, it's good technology'". ***(Ricciuti, 1994, Pg 54).***

OSF standards allow the user to decouple the application from underlying computing systems and networks so that changes made to the hardware or network does not necessitate modifications of applications. It is a vendor transparent standard, that user can mix and match hardware, software from different vendors without conflicts. ***(OSF, 2002)***

Gary Nutt came up with this definition for Open Systems "An Open System software must exhibit three properties, Interoperability, Portability, Integration and Operability". ***(Cashin, John, Pg 247, 2002)***. Each of these properties will be explained below.

- Interoperability, that refers to network protocols such as TCP/IP, Berkeley sockets.
- Portability refers to the capability of bringing across an application to a different platform and recompiling without modifying source code.
- Integration and Operability refers to ease of use, deployment, performance, and reliability of applications.

Now that we have the broad picture of OSF's role, let us look at its DCE software a bit more closely. OSF/DCE is industry standard, vendor neutral set of distributed computing technologies. It provides remote procedure call API (application programming interface) for ease of development, security services to protect and control access to data, it is scalable for organising widely scattered users and networks. OSF/DCE runs on diverse Unix-based platforms such as AIX, HP-UX, Sun's Solaris, to name but a few.

5.5.2. FreeDCE.

FreeDCE is a port of OSF/DCE for Linux-based platforms. When OSF decided to release the code for DCE, the Open Source movement was underway. This was when

around in 1998 when Jim Doyle of Boston University, rewrote major portions of the DCE code to make it work under Linux, specifically DCE's threads were rewritten and with OSF's permission to release the code under GPL. It was until around 2000, that the code base, which included Jim Doyle's work, was tidied up, with a lot of extensive renovations in terms of building the code and was available from sourceforge.net. ***(Doyle, 2002)***

Now, FreeDCE resides at *http://sourceforge.net/projects/freedce*. Unfortunately, it looks like the project has come to a standstill and it is not therefore classified as an active project. However, the software can still be downloaded.

It should be noted that I made an attempt to follow Jim Doyle's instructions on rebuilding DCE; needless to say, the whole experience wasn't satisfactory due to the number of missing but crucial bits of information required building DCE. Personally, the instructions were very poorly written, and it required a fair amount of head-scratching/hit-and-miss to get the build scripts working - which just compiles the code. But did not get very far as I realised I spent too much time trying to figure out how to get it to compile let alone to get it to work!

So I used the version available from sourceforge and the compile was painless and was easy to set-up, this surprised me! This version will be used in the implementation of the project as outlined in the next chapter.

5.6. Factors:

- *Cost Issues:*
  Since Open Source software generally has no cost in regards to licensing. This issue does not apply.
- *Ease of Use:*
  As I have mentioned, sourceforge's FreeDCE was easy to set-up, this can make certain individuals wary about this, and it all more or less depends on how experienced the individual is, again, technical mindset is mandatory and not for the faint-of-heart. Despite being easy to set-up, it is not a complete

version of DCE, as there are many things missing from it. Jim Doyle's port of OSF/DCE for Linux was certainly not easy, very poorly written that caused confusion.

- *Support/Stability:*
  Since Jim Doyle has stopped supporting his version of FreeDCE, and the FreeDCE project residing at sourceforge has come to a standstill, the question mark will be hovering over the software.

Chapter 6.

This chapter shall focus on the implementation of FreeDCE, not Jim Doyle's version, but the one from *http://sourceforge.net/projects/freedce*. I will outline the steps involved to get this version working; admittedly not very much is required. Then the implementation will be outlined and will give an overview of what was involved from the programmer's perspective in getting a Client/Server application running. The factors will be highlighted below in 6.5 covering:

- Ease of use
- Support/Stability
- Any other issues I feel are relevant.

6.1. Getting the sources.

There are two files necessary to download which constitutes the DCE package. Alas, I was in for a major disappointment after I discovered what was left after the compilation job was done, this will be mentioned here and in the next chapter as part of my conclusions. Anyway, the files required were:

- `freedce-1.1.0.7.tar.bz2`
- `dcethreads-2.0.2.tar.gz`

6.2. Building the FreeDCE package.

I will go through systematically to show what was done; the two files are located in my home directory. Commands highlighted in bold indicate what was typed at the command line.

1. Create a directory called `tmp` in my home directory

```
tommieb@tbwizards:~> mkdir tmp ↵
```

```
tommieb@tbwizards:~> cd tmp ↵
```

```
tommieb@tbwizards:~/tmp>
```

2. Unzip the files.

```
tommieb@tbwizards:~/tmp> bzip2 -dc freedce-1.1.0.7.tar.bz2|tar xvf -↵

tommieb@tbwizards:~/tmp> gzip -dc dcethreads-2.0.2.tar.gz|tar xvf -↵
```

Two directories will be created consequently, namely `freedce-1.1.0.7` and `dcethreads-2.0.2` respectively. Dcethreads needs to be built first, so that FreeDCE can compile without problems

3. Change into the `dcethreads-2.0.2` directory.

```
tommieb@tbwizards:~/tmp> cd dcethreads-2.0.2_↵
```

4. Run `configure` to create the necessary makefiles.

```
tommieb@tbwizards:~/tmp/dcethreads-2.0.2> configure ↵
```

5. Run `make` to build the dcethreads.

```
tommieb@tbwizards:~/tmp/dcethreads-2.0.2> make ↵
```

When this is one, there should be no errors. At this stage, log in as super user to install the dcethreads, this is necessary, as the install requires privileges.

6. Log in as super user.

```
tommieb@tbwizards:~/tmp/dcethreads-2.0.2> su ↵
Password: **** ↵
```

7. Run `make install` to create a common directory which will be world-readable.

```
tbwizards:/home/tommieb/tmp/dcethreads-2.0.2 # make install ↵
```

After this is done, **`/opt/dce/include`**, and **`/opt/dce/lib`** is created and this will be referenced by the FreeDCE's build process. We exit out of super user privileges, and change into the `freedce-1.1.0.7` directory.

```
tbwizards:/home/tommieb/tmp/dcethreads-2.0.2 # exit ↵

tommieb@tbwizards:~/tmp/dcethreads-2.0.2> cd ../freedce-1.1.0.7 ↵

tommieb@tbwizards:~/tmp/freedce-1.1.0.7>
```

8. Repeat steps four, five, six and seven.

After this is done, there should be a couple of binaries created in **`/opt/dce/bin`**. This is very much a common thing across all types of Open Source software, i.e.:

Run **`configure`**, run **`make`**, log on as super user, run **`make install`**.

6.3. Disappointment.

Yes, I was very disappointed after the installation of FreeDCE was performed. There was absolutely no means of setting up security, authentication, DCE cells, etc., which is a must-have when it comes to DCE development. I suppose this was to be expected, since the project came to a standstill at sourceforge.

Furthermore, I suspected that the maintainers at sourceforge did look into Jim Doyle's work and decided to abandon the security stuff, as it was overly complex to implement especially under Linux, let alone to get it to work!
My suspicions were confirmed as I sent a posting to FreeDCE's newsgroup at sourceforge, and got back a reply by a former maintainer of this FreeDCE source

code. Moreover, he mentioned that it would be a very big project involved in having to port over OSF/DCE's security over to Linux, as the code itself is quite complex. (See the appendix for more details on this posting!)

This was a big compromise in terms of the ease of building/installing FreeDCE. I hope that one day someone will come along and will want to revive the project again and hack it to make it work, one day....

6.4. Implementation.

The experiment was to see if a client/server application could work with FreeDCE. The application must be simple, concise, and demonstrate an application that would be as realistic as in the real world.

6.4.1. Requirements of client/server application.

6.4.1.1. Client application.

Present a menu interface with four options.

- Add two numbers and display the results.
- Input a string, and display each letter in the string as uppercase if the input contains lowercase letters, and to display each letter as lowercase if the input contains uppercase letters.
- Input a string and compress it using a simple run-length encoding algorithm, and to display the compressed string.

6.4.1.2. Server application.

Take in input from each of the menu options and to compute the results, send it back to the client. The server shall accept multiple clients and to process each client accordingly. Since the server demands no user interaction from the end-user, it will simply sit in the background waiting for requests coming from the client and process each request.

6.4.2. In-Depth Analysis.

The requirements are quite simple, the source code to this application can be found in the appendix. I will go through each step necessary to make it work and explain how it works. Of course, the tools that come with FreeDCE are based on the standards in terms of software development regardless of what platform is it under.

I will refrain from getting too technical here, at the same time; this analysis would be more suited to a programmer and hence look at the tools from the programmer's perspective.

6.4.2.1. IDL File.

IDL is an acronym for Interface Definition Language, and is the core of developing an RPC call. This is where passing the file through the IDL compiler generates the stub for client and server.

It is here, where the functions are made accessible both to the client and to server, and the function resides on the server via linking with server code.

This client/server development paradigm is the same regardless of whether it is commercial or Open Source DCE software. See Figure 6-1 for details.

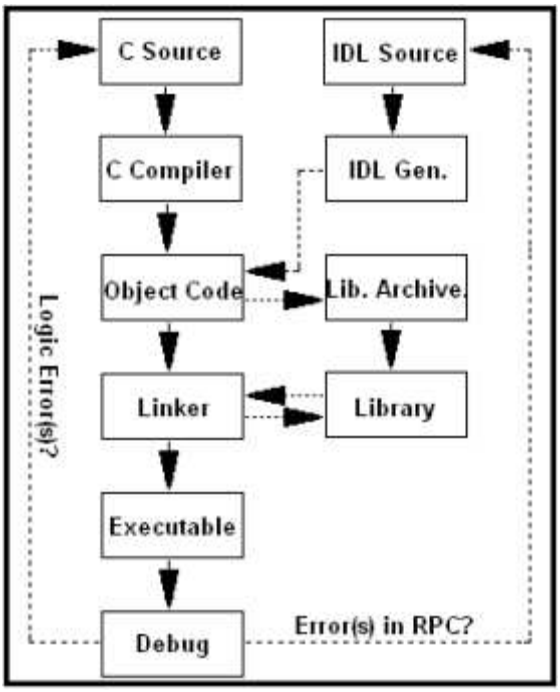

Figure 6.1. Software lifecycle with DCE.

See table 6-1 for a cut-down reproduction of an IDL file as per in the Appendix. The syntax bears resemblance to 'C' and gets parsed by a generator that creates client and server stubs.

```
/* $RCSfile: client_server.idl,v $
** $Author: tommieb $
** $Date: 2002/11/08 19:40:03 $
** $Id: client_server.idl,v 1.3 2002/11/08 19:40:03 tommieb Exp $
*/
[ uuid(98e4bfa6-f323-11d6-823e-000000000000), version(1.1), pointer_default(ptr) ]

interface tommieb
{
      typedef [ptr, string] char * string_t;

      typedef struct
      {
            unsigned32 argc;
            [size_is(argc)] string_t argv[];
      }args;

      /* My interface definition goes here */
      boolean AddIt(
            [in]        handle_t          h,
            [in]        long              in_num1,
            [in]        long              in_num2,
            [out, ref]  long              *total,
```

```
            [out, ref]  error_status_t   *status
        );
}
```

Table 6-1. An IDL file containing simple RPC function called AddIt.

In the above sample of an IDL file, an RPC function is defined and called **`AddIt`** which takes in a number of parameters, in respective order,

- The handle to the RPC binding variable.
- The first number to pass in.
- The second number to pass in.
- A reference variable holding the sum of the two numbers.
- A reference variable containing the error of the RPC should it fail?

The output files (client/server stubs) are in machine code native to the platform; this is linked with the other code to produce a final binary executable ready for execution.

As shown, in figure 6-1, the complexity of the development lifecycle is proportional to the complexity of the project is. It may sound contradictory if in terms of compiling the code via make files which does make life easier, having said that, it is the maintenance part such as debugging/testing that can get quite complex. As in this case for the simple implementation, it was quite simple.

To quote, "Development of the distributed application varies with computing environment, individual skill and knowledge". ***(Chu, 1997, Pg. 268)*** "Code development style for distributed application is quite different from conventional approach, leading the cost increase for distributed applications development" ***(Chu, 1997, Pg. 278)***

His paper on the code development makes a very valid point, as the coding approach **is** different. In the case of the implementation, there was an exception handler that is very similar to C++'s **`try/catch`** exception handler that is based on complex C macros via **`setjmp(…)`** and **`longjmp(…)`**, and took a bit of getting used to even from the perspective of C.

Had Jim Doyle's port of OSF/DCE for Linux worked, there would have been a few idiosyncrasies in setting up the cells, and that would require extensive knowledge in areas of troubleshooting and error reporting.

6.5. Factors governing Implementation using Open Source.

- *Cost Issues:*
  I have mentioned in the previous chapter regarding costs in licencing.
- *Ease of Use:*
  As I have mentioned, sourceforge's FreeDCE was easy to set-up, this can make certain individuals wary about this, and it all more or less depends on how experienced the individual is, again, technical mindset is mandatory and not for the faint-of-heart. Despite being easy to set-up, it is not a complete version of DCE, as there are many things missing from it.
- *Support/Stability:*
  Since Jim Doyle has stopped supporting his version of FreeDCE, and the FreeDCE project residing at sourceforge has come to a standstill. The question mark will be hovering over the future of this software that is unless someone is brave enough to examine OSF/DCE's code to make it work.

Chapter 7.

This chapter wraps up the conclusions and any issue(s), surrounding this dissertation. I will go through the following factors:

- Cost, this will govern both hardware and software, also known as TCO, from Case study perspective, Commercial and Open Source perspective as well.
- Ease of Use.
- Support/Stability.

I will give a brief introduction into TCO, and then cover TCO in terms of hardware and software. As per in the case study as outlined in Chapter 3, the cost of Entera is unknown for the reason that Borland would not disclose a quote. However, I will try to use a number of variables as outlined in Chapter 4. Entegrity willingly supplied the quote on the basis that the software will be running under two distribution of Linux – RedHat and SuSE. The quotation can be found in the appendix. I will start of by explaining TCO and how it works.

7.1. Brief introduction to TCO.

TCO or Total Cost Ownership is an assessment factor that is taken into consideration by many organisations. It enables to understand the expenditure or cost of owning, producing and using an IT entity (hardware or software). The simplest way of knowing the cost is by using simple maths to calculate the sum. There can be unforeseen factors that can affect it such as an upgrade of hardware, software, contingency plans such as prevention or loss of data, IT management planning and mindset, to name but a few. For some organisations that are conscious of their budget and the need to know where it is going, this can help them see what direction to take and to make a balanced decision on this basis. That is the positive aspect of a TCO study.

Here I will outline a number of factor(s) that would be perceived as the driving force behind TCO:

- Platform(s) – Windows, *nix, MacOS.
- Architecture(s) – such as Intel x86, RISC.
- IT Management – such as poor communication/management techniques, inadequate investments in hardware/software, inadequate staffing levels etc.
- Support – Poor response time, lack of training/awareness etc.

7.2. Estimated cost from Case Study.

Let us look at the cost of the hardware mentioned in the case study. I will use the following specifications for entry-level workstation supplied by IBM as a guideline for this basis.

7.2.1. Hardware.

IBM RS/6000 Entry Level pSeries 610 Model 6E1 Express.

Specifications:

- 64bit architecture.
- 450 MHz Dual processor.
- 8Mb Level two Cache.
- 2 x 36 GB RAID.
- 2 GB RAM.
- 5 PCI slots.
- 3 serial and 2 parallel ports
- Two integrated Ethernet NIC 10/100Mbps.
- AIX v4.3.3

Total cost of hardware *(H)*: $17,395.

Upgrades etc is not accounted for in this TCO.
The cost of this workstation would be seen as a once off as it is looked at as an investment or capital.

7.2.2. Software.

I have looked at in Chapter 4, what would be the cost of the Entera software. I have duplicated it here to make this chapter feel complete.

Licence cost for developers = 50 x Single_Developers_Cost_Of_Licence ***(A)***
Licence cost for end-users = NoClients x Single_Client_Cost_Of_Licence ***(B)***
On-site support of hardware/software from IGS = Z x Cost per Year ***(Z)***
On-site software support (Borland Personnel go on-site) = Y x Cost per Year ***(Y)***
Technical Support by Borland = S x Cost Per Year ***(S)***

Total cost of software *(T)*: (*A*) + (*B*) + (*Z*) + (*Y*) + (*S*)

Note: *A* and ***B*** would be considered to be once off. However, for ***Y***, usually, Borland would offer a discount for support for say three/five years (depending on how long they would support the software for)

Any additional licences such as expansion of development team is not taken into consideration here, but as you can see in the simple maths above, this can work out to be quite expensive on a yearly basis!

7.2.3. Conclusions from Case Study.

7.2.3.1. Cost.

Putting the two sums in the above, together; notice that ***H*** is once off; it still works out to be quite expensive:

Overall Total Expenditure for Entera and IBM pSeries System: $(*H* + *T*)

Obviously, that can be perceived to be high, in terms of operational costs or running costs of keeping the project alive. Hence, management was willing to trim down the expenses by investing into Java and JavaBeans; still, even somewhat, the amount saved would be made up for the cost of cross-training individuals to Java.

7.2.3.2. Ease of Use.

The process of hiding the low-level nuts and bolts of the software where conventional software development is concerned – is very good, the development team are not put off the beaten track and are able to maintain their focus on getting the logic working.

I will repeat this here from the previous chapter, "Code development style for distributed application is quite different from conventional approach, leading the cost increase for distributed applications development" (***Chu, 1997, Pg. 278).*** It would be a reasonable assumption that Borland did a very good job in the development of Entera. As the software team, in which I was a participant of, did not lose productivity or momentum and hence were able to get the job done. There was no need to worry about the technical details as this was all hidden, I can assume that Entera must have been a very expensive package put together to make everyday programmer's lives easier.

Also, there are "hands on decks", by IGS and Borland. They were able to fine-tune the software behind the scenes. I am conscious of the fact that I was talking about it from the programmer's perspective, speaking for the developers here.

7.2.3.3. Support/Stability

Putting cost aside for a moment, the organisation in the case study was willing to pay the price initially – question is why? Security and support, the management's mindset was at ease in terms of feeling secure that there are experts who are backing up the hardware/software in case of unforeseen problems. There is IGS and Borland who are willing to give a hand and examine what went wrong, in event of a situation occurring that requires expertise and knowledge.

7.3. Commercial Software – Entegrity.

Let us look at it from Entegrity's perspective with Linux as the system installed.

Licence cost for developers = 50 x $110 ***(A, $5,500)***

- **This will be likely to change once Entegrity markets a server version!**

Licence cost for end-users = NoClients x Single_Client_Cost_Of_Licence ***(B)***

- **Hard to estimate as Entegrity develops for a number of platforms such Windows, Linux, and hence cost could be variable depending on end-user's platform in question.**

Cost of buying boxed version of Linux Distribution e.g. SuSE 7.3 ***(Circa $50-$60)***

- **You can install it on as many systems as required; therefore, licence cost is $0!**

On-site support of Entegrity = Z x Cost per Year ***(Z)***

- **An organisation can have two choices here; hire an experienced/competent Systems Administrator (*will be cheaper in the long run!*). Or get Entegrity to send out qualified personnel specialist who would charge by the hour or similar contract. (*Expensive!*)**

On-site software support = Y x Cost per Year ***(Y)***

- **Ditto *Z*.**

Technical Support by Entegrity = 20% of A ***(S, $5,500 x 20% = $1,100)***

Total cost of software *(T)*: (***A***) + (***B***) + (***Z***?) + (***Y***?) + (***S***).

Again, ***A*** and ***B*** would be once off. ***Z*** and ***Y*** could be eliminated from the maths above if an organisation hires a Systems Administrator. So we're looking at a figure of $1,100 + ***B***. Sounds reasonable enough provided that the cost of a single licence does not jump much, once a server version gets to be released in the near future.

7.3.1. Conclusions about Entegrity.

This is an interesting company to watch out for; we have to remember that their DCE software was a derivative of OSF/DCE. Once they get the server version out; it will open the doors of opportunity for organisations to make the switch over to Linux, moreover, the capability to migrate existing applications across from legacy systems.

7.3.1.1. Cost.

The cost is an interesting one and all people who are familiar with DCE are keeping a look-out for this, even then, with the licence cost of Linux to be $0, there still would be money to be saved. Their evaluation version that is available for download, which I think, is a very good idea thus enabling developers to try it out for thirty days. There is a caveat emptor with the evaluation version that might tempt developers or organisations to try it.

Would developers who downloaded it, be willing to spend a few days in setting up a Linux box and taking a small subset of their application or organisation's, and integrate it with Entegrity to try it out? There are hidden costs in establishing whether that would work or not for them. So, despite the evaluation being a good idea, one must not forget in planning before downloading the software!

7.3.1.2. Ease of use and Support/Stability.

Due to the fact, that I was unable to use it, as it required existing DCE cells/brokers up and running, which I do not possess. I was therefore unable to determine how easy it would be to use and how stable it would be. However, it must not be forgotten or ignored, that the stability of Linux might even lull a false sense of security, "Hey, this DCE software is really stable?" or "This Linux distribution is very stable?"

7.4. Open Source Software – FreeDCE.

Licence cost for developers = 50 x Single_Developers_Cost_Of_Licence ***(A $0)***
Licence cost for end-users = NoClients x Single_Client_Cost_Of_Licence ***(B $0)***
Cost of buying boxed version of Linux Distribution e.g. SuSE 7.3 ***(Circa $50-$60)***

- **You can install it on as many systems as required; therefore, licence cost is $0!**

On-site support of Boxes = Z x Cost per Year ***(Z)***
On-site software support = Y x Cost per Year ***(Y)***
Technical Support = S x Cost Per Year ***(S)***

- **For Z, Y, S, from an organisation's perspective, their only choice is to hire an experienced/competent Systems Administrator (*will be cheaper in the long run!*), who shares enthusiasm of Open Source. Yet, one must not forget the cost of dialling-up/spending time on the Internet and help forums searching for answers/patches/fixes if he/she is willing to do that.**

7.4.1. Conclusions about Open Source - FreeDCE.

7.4.1.1. Cost.

There are no costs in regards to Open Source. Though I must admit, I am very curious at how Entegrity were able to market their product, since OSF open sourced their DCE software for academic/educational use only; there are a number of speculative questions arising from it. Did Entegrity buy out the complete source, modified it for Linux to the tune of $12M as per OSF's Extended DCE Licensing Agreement?[8] Alternatively, did they "borrow" the open source code, modify it, and sell it on? *This is highly unlikely as it would violate OSF's licence or breach of contract and thus Entegrity would end up in a legal wrangle or even worse, sued by OSF.*

[8] For more information, *http://www.opengroup.org/tech/dce/info/dce122pricelist.htm*

There is a very valid question that I feel should be mentioned here:

- **Should Entegrity open source their DCE software since it is available for Linux and other platforms, and thus Linux is GPL'd, therefore software available for a GPL'd system should also be GPL'd?**

Now, the complexity behind the GPL licence can be seen here and must be approached very carefully!

7.4.1.2. Ease of Use.

I will mention OSF/DCE's Jim Doyle port for Linux. It was a complete nightmare as I have highlighted in Chapter 5. It would be fair to say; never ever again will I play about in setting it up due to poor documentation, ambiguous statements etc. Furthermore, the structure of the source code tree was made a bit messy thanks to following the documentation! Let us look at it from sourceforge's FreeDCE perspective.

The ease of use can be a bit ambiguous here; are we talking about in terms of setting it up such as doing a `configure`, `make`, `make install` as I covered in the previous chapter? On the other hand, is it in terms of programmer's perspective?

I will provide the two perspectives; I have been an advocate of Open Source for a long time since I first obtained Linux Slackware 3.0 (October 1995). In addition, I am quite used to doing the procedure in setting it up and it gets easier each time and more rightly, it is a standard way of setting up the binaries once source code is downloaded. There is a hidden bonus in doing it this way; the source code is compiled natively depending on the specifications of the machine and is optimised for that processor.

From the programmer's perspective, the setting up of the implementation was; what I thought was a bit fiddly, for instance, the IDL compiler that came with it was a tedious process. It turned out that the utility `uuid` had to be executed initially, in order to generate a unique number. By redirecting the output from the utility to a file that

had to be edited manually to include the RPC functions definition. Then this edited file had to be passed through the **`dceidl`** compiler to generate client/server stubs. Very messy indeed, with makefiles, the process would be painless without having to remember the certain parameters supplied with switches on the command line.

The documentation was poor; at one stage, I got a very strange error message with no meaningful statement or action on what to do. The error message indicated a number in hexadecimal format – "**`0x16c9a01e`**" and thus not very friendly or informative. I ended up spending two days trying to track down the error. Until I came across the constants for the numbers in a header file in the **`include`** directory via searching (in Unix speak, **`grep`**'ing) the headers for the constant, which meant the "RPC server was already registered". I had to manually un-register the server by explicitly calling **`rpc_server_unregister_if(…)`**, and everything was fine after that.

The FreeDCE software in general lacks a lot of functionality to make a distributed environment and is inactive or that there is not much interest in keeping the project alive as of now.

7.4.1.3. Support/Stability

I have illustrated the point about Open Source - there is no such thing as a general company specialising in supporting Linux or Open Source for that matter. There is an exception though; some distribution companies such as RedHat, SuSE do offer support for 90 days, this can be extended with special deals/contracts at a cost; **only for problems with an installation of Linux**. Any other installation of such software that is not recognised by the distribution companies are not covered, which is fair enough as it is not their responsibility.

There is an old adage about using Open Source, and it is known as **RTFM** or **R**ead **T**he **F**ine **M**anual, and it is funny that documentation is one of the weakest points about Open Source. The exclusion of boxed distributions applies here since they come with very good manuals; this explains why they are sold around $50-$60 per box. Of course, the reason is that programmers collaborating from vast corners of

the world are involved in projects. Language barriers have yet to be broken which is the major factor behind the poor quality of documentation. Notice that this does not imply as a bad thing, as documentation would be available in their native tongue, which would be seen as a good thing. Despite the English language being the most popular, there are some programmers who have not the basics, hence, the language barriers.

Nonetheless, there are thousands of newsgroups, site forums, where individuals post their query and there is a very high chance that the postings would be answered. In addition, since I have illustrated in Chapter 5, Open Source software tends to be more robust, have responsive turn-around time concerning patches/fixes shortly after release. And furthermore, by searching newsgroups, site forums, it is surprising how one would gain knowledge from it and that far outweighs any negligible costs in terms of dialling up and spending time online searching for answers!

7.5. Overall Conclusions.

Open Source sourceforge's FreeDCE was a disappointment, there was no functionality, and was hoping initially that it would have capabilities of setting up cells with security authentication. Alas that is not the case, due to the complexity of the code behind the OSF/DCE which was only portable to the Apollo, HP800, AIX, SVR4 and i386_sinix Unix variants.

Had there been more involvement with the project, there would have been more functionality/features. Despite Jim Doyle who did a lot of work in porting it across to the Linux platform, it is sad to see that his efforts for cell setting up, authentication et al, did not make its way to sourceforge.

Entegrity, by far, looks the most promising, who knows, it may release its code under Open Source. As the momentum is going from strength to strength, chances are many software companies will release it under a somewhat restricted version of GPL in order to beat off competition in the market place. Of course, the market to stay on top only depends on whether the software companies have courage and to show the world what their code is made of.

I would most certainly keep an eye on this promising aspect, unfortunately I am not a clairvoyant so I will refrain from making any predictions here, but this is one company to watch out.

Borland's Entera, by far, the superior of the lot, holds tremendous staying power when it comes to the DCE technology, but it comes with a price. Ease of use, focus on getting the work done, abstraction of technical level hidden away from programmers, clever handling of servers, this is a very polished, mature product.

Unfortunately, CORBA, WebSphere, MQSeries are maintaining their stance in today's software development trends and have shown to be far superior then DCE. Again, this will depend on the management's mindset on whether migrate the existing DCE legacy applications to the more modern technology. While looking for ways in keeping the cost down, and is highly dependent on their nature of business requirements.

Open Source is a viable alternative despite weakness in documentation. There will be opportunities for somebody who will start up a project. In addition, if it is perceived to be a good idea in terms of "We could do with this"; no doubt, development interest will rise. As I have demonstrated above that the licence cost is the major factor in terms of saving money. Despite FreeDCE having their weaknesses, and with projects climbing upwards, more programmers are donating their time in development. With big companies willing to spend money on development, research and design and releasing proprietary code in order to stay one step ahead, the future looks promising and bright.

Appendix A - GPL Licence.

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you". Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty;

and give any other recipients of the Program a copy of this License along with the Program.
You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.
Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.
In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.
If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt

otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>
Copyright (C) 19yy <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.
If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.
You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix B - Source Code for implementation using Sockets.

File: Makefile

```
.SUFFIXES: .c.o
CC = gcc
INCLUDE = -I/usr/include
LIBS = -L/usr/lib -lncurses
CFLAGS = -g #-Wall -W -pipe -g #-Werror -O

CLIENT_FILES = menu_client.o main_client.o
CLIENT_APP = client_app

SERVER_FILES = main_server.o
SERVER_APP = server_app

all: $(CLIENT_APP) $(SERVER_APP)

.c.o:
    $(CC) $(INCLUDE) $(CFLAGS) -c $<

$(CLIENT_APP): $(CLIENT_FILES)
    $(CC) -o $@ $(CFLAGS) $(INCLUDE) $(LIBS) $(CLIENT_FILES)

$(SERVER_APP): $(SERVER_FILES)
    $(CC) -o $@ $(CFLAGS) $(INCLUDE) $(LIBS) $(SERVER_FILES)

clobber:
    rm -rf $(CLIENT_FILES) $(SERVER_FILES)

clean:
    rm -rf $(CLIENT_APP) $(SERVER_APP)
```

File: in_out_data.h

```
/*
** File: "$RCSfile: in_out_data.h,v $"
** Author: "$Author: tommieb $"
** Date: "$Date: 2003/01/19 17:39:12 $"
** Purpose: Arrangement for data marshalling/unmarshalling buffer.
*/

#ifndef __IN_OUT_DATA_H
#define __IN_OUT_DATA_H

#define IN_OUT_DATA_SZ        80

char in_out_data[IN_OUT_DATA_SZ];

#endif /* __IN_OUT_DATA_H */

File: main_client.h
/*
** File: "$RCSfile: main_client.h,v $"
** Author: "$Author: tommieb $"
** Date: "$Date: 2003/01/19 17:39:12 $"
** Purpose: Header file for client side.
*/

#ifndef __MAIN_CLIENT_H
#define __MAIN_CLIENT_H

#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <getopt.h>
#include <unistd.h>
#include <curses.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#ifdef __cplusplus
```

```
extern "C" {
#endif

static char *RCSIdHdr __attribute__((__unused__)) = "$Id: main_client.h,v 1.1
2003/01/19 17:39:12 tommieb Exp tommieb $";
extern int verbose, fdnet;

/* main_client.c */
void cleanup(void);
void panic(const char *, ...);
void init_signals(void);

/* menu_client.c */
int draw_menu(void);
int process_two_nums(void);
void process_string(int);

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_CLIENT_H */

File: menu_client.c
/*
** File: "$RCSfile: menu_client.c,v $"
** Author: "$Author: tommieb $"
** Date: "$Date: 2003/01/19 18:00:20 $"
** Purpose: This is where the menu options are processed...
*/

#include "main_client.h"
#include "in_out_data.h"

static char *Id __attribute__((__unused__)) = "$Id: menu_client.c,v 1.2 2003/01/19
18:00:20 tommieb Exp tommieb $";

int draw_menu(void)
{
    char str_num[2];
    int num, valid_num = 0;
    mvprintw(3, 2, "----> Client <----");
    mvprintw(5, 2, "1. Add two numbers.\n\n");
    mvprintw(7, 2, "2. Input a string.\n\n");
    mvprintw(9, 2, "3. Compress a string.\n\n");
    mvprintw(11, 2, "9. Goodbye!\n");
    refresh();
    crmode();
    do
    {
        mvprintw(13, 2, "Enter Choice (9 to Quit): ");
        refresh();
        clrtoeol();
        getnstr(str_num, 2);
        sscanf(str_num, "%d", &num);
        if ((num >= 1) && (num <= 9))
            valid_num = 1;
    }while (!valid_num);
    return num;
}

int process_two_nums(void)
{
    char str_num[2];
    int valid_num = 0, quit = 0, value, rv;
    long num1, num2, Total;
    clear();
    mvprintw(3, 2, "----> Client. (1. Add two numbers) <----");
    refresh();
    nocrmode();
    do
    {
        mvprintw(5, 2, "Enter first number (0 to exit): ");
        refresh();
        clrtoeol();
        getnstr(str_num, 2);
        rv = sscanf(str_num, "%ld", &num1);
```

```
        if (num1 == 0 || rv == 0)
            quit = 1;
        else
            valid_num = 1;
    }while (!valid_num);
    if (quit)
    {
        mvprintw(7, 2, "User quitted!");
        crmode();
        getch();
        return 1;
    }
    valid_num = 0;
    nocrmode();
    do
    {
        mvprintw(7, 2, "Enter second number (0 to exit): ");
        getnstr(str_num, 2);
        clrtoeol();
        rv = sscanf(str_num, "%ld", &num2);
        if (num2 == 0 || rv == 0)
            quit = 1;
        else
            valid_num = 1;
    }while (!valid_num);
    if (quit)
    {
        crmode();
        mvprintw(9, 2, "User quitted!");
        getch();
        return 1;
    }
    if (verbose)
        /* Call the server! */
        mvprintw(9, 2, "Calling the server....");
    sprintf(in_out_data, "1;%d,%d", num1, num2);
    write(fdnet, in_out_data, sizeof(in_out_data));
    read(fdnet, &in_out_data, sizeof(in_out_data));
    sscanf(in_out_data, "%d;%d", &value, &Total);
    if (value == 100)
        if (verbose)
            mvprintw(11, 2, "Got result from server....%ld", Total);
        else
            mvprintw(11, 2, "Result is %ld", Total);
    refresh();
    crmode();
    mvprintw(13, 25, "Press a key...");
    getch();
    return 0;
}

void process_string(int compress)
{
#define MY_MAX_INPUT   30
    char str_inp[MY_MAX_INPUT], *ptr;
    int value;
    clear();
    nocrmode();
    if (!compress)
        mvprintw(3, 2, "----> Client. (2. Input a string) <----");
    else
        mvprintw(3, 2, "----> Client. (3. Compress a string) <----");
    mvprintw(5, 2, "Enter a string (No entry to exit): ");
    getnstr(str_inp, sizeof(str_inp));
    if (!strlen(str_inp))
    {
        crmode();
        mvprintw(7, 2, "User quitted!");
        getch();
        return;
    }
    if (verbose)
        mvprintw(7, 2, "Calling the server....");
    if (!compress)
        sprintf(in_out_data, "2;%s", str_inp);
    else
        sprintf(in_out_data, "3;%s", str_inp);
```

```
    ptr = strdup(str_inp);
    write(fdnet, in_out_data, sizeof(in_out_data));
    read(fdnet, &in_out_data, sizeof(in_out_data));
    sscanf(in_out_data, "%d;%s", &value, str_inp);
    switch(value)
    {
        case 102    :    mvprintw(9, 2, "Original string was %s and length was %d",
ptr, strlen(ptr));
                         if (verbose)
                             mvprintw(11, 2, "Got result from server....");
                         mvprintw(12, 14, "Compressed string is %s and length is %d",
str_inp, strlen(str_inp));
                         break;
        case 101    :    ptr = (char *) &in_out_data;
                         while (*ptr != ';') *ptr++;
                         *ptr++;
                         if (verbose)
                             mvprintw(9, 2, "Got result from server....%s", ptr);
                         mvprintw(9, 2, "Result is %s", ptr);
                         break;
    }
    refresh();
    crmode();
    mvprintw(14, 25, "Press a key...");
    getch();
    return;
}
```

File: main_client.c

```
/*
** File: "$RCSfile: main_client.c,v $"
** Author: "$Author: tommieb $"
** Date: "$Date: 2003/01/19 17:39:12 $"
** Purpose: Demonstrates a menu driven interface!
*/

#include "main_client.h"
#include "in_out_data.h"

/* RCS Ident. */
static char *RCSId __attribute__((__unused__)) = "$Id: main_client.c,v 1.1 2003/01/19
17:39:12 tommieb Exp tommieb $";

/* Global variables pertaining to command line options */
int verbose = 0, curses_init = 0;
char tcp_host[128] = "localhost"; /* default! */
int fdnet;
/* Signal handler struct */
struct sigaction sigact;

/* Functions */
static void signal_handler(int);
static void shutdown_client(void);
static void usage(void);
static int tcp_connect(const char *, int);

int main(int argc, char **argv)
{
    extern char *optarg;
    extern int optind, opterr, optopt;
    int c, menu_done = 1, menu_opt, arg_port = -1;

    /* Process command line! */
    while ((c = getopt(argc, argv, "h:p:v")) != EOF)
    {
        switch (c)
        {
            case 'v'    :    verbose = 1;
                             break;
            case 'p'    :    /* Grab port # */
                             arg_port = atoi(optarg);
                             break;
            case 'h'    :    strncpy(tcp_host, optarg, sizeof(tcp_host)-1);
                             break;
            default     :    usage();
        }
```

```
    }

    /* Initialise curses */
    initscr();
    curses_init = 1;
    /* Set kbd to cooked mode */
    nocbreak();
    /* Set up signal handler! */
    init_signals();
    /* At first sign of trouble, we cleanup! */
    atexit(cleanup);

    if (arg_port == -1)
        arg_port = 9876;
    /* Connect to the server! */
    fdnet = tcp_connect(tcp_host, arg_port);
    /* Do the biz! */
    while (menu_done)
    {
        clear();
        menu_opt = draw_menu();
        switch(menu_opt)
        {
            case 1    :    process_two_nums();
                           break;
            case 2    :    process_string(0);
                           break;
            case 3    :    process_string(1);
                           break;
            case 9    :    clear();
                           endwin();
                           curses_init = 0;
                           menu_done = 0;
                           break;
            default    :    break;
        }
    }
    exit(EXIT_SUCCESS);
}

void panic(const char *fmt, ...)
{
    va_list ap;
    if (curses_init)
    {
        endwin();
        curses_init = 0;
    }
    va_start(ap, fmt);
    vfprintf(stderr, fmt, ap);
    va_end(ap);
    exit(EXIT_FAILURE);
}

void init_signals(void)
{
    sigact.sa_handler = signal_handler;
    sigemptyset(&sigact.sa_mask);
    sigact.sa_flags = 0;
    /* Did we get ctrl-c'd */
    sigaction(SIGINT, &sigact, (struct sigaction *) NULL);
    /* Did we bomb? */
    sigaddset(&sigact.sa_mask, SIGSEGV);
    sigaction(SIGSEGV, &sigact, (struct sigaction *) NULL);
}

static void signal_handler(int sig)
{
    if (sig == SIGINT)
        panic("Whoops! We got ^c'd....\n");
    if (sig == SIGSEGV)
        panic("Whoops! We blew up....\n");
}

void cleanup(void)
{
    /* Reset Signal handlers */
```

```
    sigact.sa_flags = SA_RESETHAND;
    sigaction(SIGINT, &sigact, 0);
    sigaction(SIGSEGV, &sigact, 0);
    /* Shutdown client gracefully! */
    if (curses_init)
    {
        endwin();
        curses_init = 0;
    }
    shutdown_client();
}

static void shutdown_client(void)
{
    close(fdnet);
}

static void usage(void)
{
    printf("usage:  client [-h hostname] [-p] [-v]\n");
    printf("        -h : specify host where DCE server lives\n");
    printf("        -p : use port number\n");
    printf("        -v : verbose mode\n");
    exit(1);
}

int tcp_connect(const char *hostname, int port)
{
    struct hostent *host_info;
    struct sockaddr_in server;
    int sock;
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock < 0)
        return -1;
    host_info = gethostbyname(hostname);
    server.sin_family = AF_INET;
    memcpy((char *) &server.sin_addr, host_info->h_addr, host_info->h_length);
    server.sin_port = htons(port);
    connect(sock, (struct sockaddr *) &server, sizeof(server));
    return sock;
}
```

File: main_server.c

```
/*
** File: "$RCSfile: main_server.c,v $"
** Author: "$Author: tommieb $"
** Date: "$Date: 2003/01/19 18:00:20 $"
** Purpose: The server code!
*/

#include <stdio.h>
#include <signal.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdarg.h>
#include "in_out_data.h"

#define LISTEN_PORT           9876

/* RCS Ident. */
static char *RCSId __attribute__((__unused__)) = "$Id: main_server.c,v 1.2 2003/01/19
18:00:20 tommieb Exp tommieb $";

/* Signal handler struct */
struct sigaction sigact;

/* Global variables */
int fdnet;

/* Functions */
static void init_signals(void);
static void signal_handler(int);
```

```
static void shutdown_server(void);
static void usage(void);
static void panic(const char *fmt, ...);
static void cleanup(void);
int rle_compress(const char *, char **, int);
int create_tcp_endpoint(int);

int main(int argc __attribute__((__unused__)), char **argv
__attribute__((__unused__)))
{
    int value, sock, total, num1, num2;
    char *ptr, tmp_buf[40], *tmp_buf_ptr1, *tmp_buf_ptr2;

    /* Set up signal handler! */
    init_signals();
    /* At first sign of trouble, we cleanup! */
    atexit(cleanup);

    if ((sock = create_tcp_endpoint(LISTEN_PORT)) < 0)
        panic("Unable to establish listening endpoint!\n");
    /* Get connection from client! */
    fdnet = accept(sock, NULL, NULL);
    while (1)
    {
        /* Read in the incoming marshalled data */
        read(fdnet, in_out_data, sizeof(in_out_data));
        printf("Marshalled data read in: %s\n", in_out_data);
        /* Unmarshall and process the data */
        ptr = (char *) &in_out_data;
        switch (*ptr++)
        {
            case '1'    :    /* Add two numbers */
                            *ptr++;
                            sscanf(ptr, "%d,%d", &num1, &num2);
                            total = num1 + num2;
                            /* Clear the buffer */
                            in_out_data[0] = '\0';
                            sprintf(in_out_data, "100;%d", total);
                            break;
            case '2'    :    /* string input */
                            *ptr++;
                            tmp_buf_ptr1 = (char *)&tmp_buf;
                            /* Do the business */
                            while (*ptr)
                            {
                                if (isupper(*ptr) && isalpha(*ptr))
                                    *tmp_buf_ptr1 = tolower(*ptr);
                                else
                                    if (islower(*ptr) && isalpha(*ptr))
                                        *tmp_buf_ptr1 = toupper(*ptr);
                                    else
                                        *tmp_buf_ptr1 = *ptr;
                                    *tmp_buf_ptr1++; *ptr++;
                            }
                            *tmp_buf_ptr1 = '\0';
                            /* Marshall the new values - Clear the buffer */
                            in_out_data[0] = '\0';
                            sprintf(in_out_data, "101;%s", tmp_buf);
                            break;
            case '3'    :    /* compress a string */
                            *ptr++;
                            /* string input */
                            tmp_buf_ptr1 = (char *)&tmp_buf;
                            /* Copy the input string to temporary buffer */
                            while (*ptr)
                                *tmp_buf_ptr1++ = *ptr++;
                            *tmp_buf_ptr1 = '\0';
                            tmp_buf_ptr2 = (char *)&tmp_buf;
                            tmp_buf_ptr2 = strdup(tmp_buf);
                            /* Do the business! */
                            if (rle_compress(tmp_buf, &tmp_buf_ptr2, strlen(tmp_buf) +
1) == 2)
                                sprintf(in_out_data, "102;%"); /* Corruption occurred
- notify client! */
                            /* Clear the buffer */
                            in_out_data[0] = '\0';
                            sprintf(in_out_data, "102;%s", tmp_buf_ptr2);
```

```
                            break;
        }
        if (value == 9)
            break;
        /* Pass the marshalled data back to the client! */
        write(fdnet, in_out_data, sizeof(in_out_data));
    }
    close(fdnet);
    exit(EXIT_SUCCESS);
}

int create_tcp_endpoint(int port)
{
    int sock;
    struct sockaddr_in server;
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock < 0)
        return -1;
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = htonl(INADDR_ANY);
    server.sin_port = htons(port);
    if (bind(sock, (struct sockaddr *) &server, sizeof(server)) < 0)
        return -2;
    listen(sock, 5);
    return sock;
}

void panic(const char *fmt, ...)
{
    va_list ap;
    va_start(ap, fmt);
    vfprintf(stderr, fmt, ap);
    va_end(ap);
    exit(EXIT_FAILURE);
}

static void init_signals(void)
{
    sigact.sa_handler = signal_handler;
    sigemptyset(&sigact.sa_mask);
    sigact.sa_flags = 0;
    /* Did we get ctrl-c'd */
    sigaction(SIGINT, &sigact, (struct sigaction *) NULL);
    /* Did we bomb? */
    sigaddset(&sigact.sa_mask, SIGSEGV);
    sigaction(SIGSEGV, &sigact, (struct sigaction *) NULL);
}

static void signal_handler(int sig)
{
    if (sig == SIGINT)
        panic("Whoops! We got ^c'd....\n");
    if (sig == SIGSEGV)
        panic("Whoops! We blew up....\n");
}

void cleanup(void)
{
    /* Reset Signal handlers */
    sigact.sa_flags = SA_RESETHAND;
    sigaction(SIGINT, &sigact, 0);
    sigaction(SIGSEGV, &sigact, 0);
    /* Shutdown server gracefully! */
    shutdown_server();
}

static void shutdown_server(void)
{
    close(fdnet);
}

int rle_compress(const char *orig_data, char **compressed_data, int orig_data_len)
{
    int match = 1, first_time = 1;
    char save_orig_char = 'A', *ptr_to_orig_data, *ptr_to_comp_data,
*compressed_data_buf;
```

```
    if (!(compressed_data_buf = (char *) malloc(sizeof(char) * (orig_data_len + 5))))
        return 2;
    ptr_to_comp_data = compressed_data_buf;
    /* Signature! */
    *ptr_to_comp_data++ = 0xBE; *ptr_to_comp_data++ = 0xEF;
    for (ptr_to_orig_data = (char *)orig_data; *ptr_to_orig_data; *ptr_to_orig_data++)
    {
        if (first_time)
        {
            first_time = 0;
            save_orig_char = *ptr_to_orig_data;
        }
        else
        {
            if (*ptr_to_orig_data == save_orig_char)
                match++;
            else
            {
                *ptr_to_comp_data++ = match;
                *ptr_to_comp_data++ = (char)save_orig_char;
                save_orig_char = *ptr_to_orig_data;
                match = 1;
            }
        }
    }
    *ptr_to_comp_data++ = match;
    *ptr_to_comp_data++ = (char)save_orig_char;
    save_orig_char = *ptr_to_orig_data;
    *ptr_to_comp_data = '\0';
    (*compressed_data) = compressed_data_buf;
    return(1);
}
```

Appendix C - Source code for implementation using FreeDCE.

File: Makefile

```
.SUFFIXES: .c.o
CC = gcc
IDL = /opt/dce/bin/idl
INCLUDE = -I/opt/dce/include -I/usr/include
LIBS = -L/opt/dce/lib -L/usr/lib -ldcerpc -luuid -lstdc++ -lncurses
CFLAGS = -D_REENTRANT -D_GNU_SOURCE -Wall -W -pipe #-Werror -O
IDL_GEN_HDR = mc.h
IDL_FLAGS = -header $(IDL_GEN_HDR)

CLIENT_FILES = menu_client.o main_client.o dce_misc.o
CLIENT_IDL = client_server.idl
CLIENT_STUB = client_server_cstub.o
CLIENT_APP = client_app

SERVER_FILES = main_server.o dce_misc.o
SERVER_STUB = client_server_sstub.o
SERVER_APP = server_app

all: $(CLIENT_APP) $(SERVER_APP)

.c.o:
      $(CC) $(INCLUDE) $(CFLAGS) -c $<

$(CLIENT_STUB): $(CLIENT_IDL)
      $(IDL) $(IDL_FLAGS) $(INCLUDE) $(CLIENT_IDL)

$(CLIENT_APP): $(CLIENT_STUB) $(CLIENT_FILES)
      $(CC) -o $@ $(CFLAGS) $(INCLUDE) $(LIBS) $(CLIENT_FILES) $(CLIENT_STUB)

$(SERVER_APP): $(SERVER_STUB) $(SERVER_FILES)
      $(CC) -o $@ $(CFLAGS) $(INCLUDE) $(LIBS) $(SERVER_STUB) $(SERVER_FILES)

clobber:
      rm -rf $(CLIENT_STUB) $(SERVER_STUB) $(CLIENT_FILES) $(SERVER_FILES)
$(IDL_GEN_HDR)

clean:
      rm -rf $(CLIENT_APP) $(SERVER_APP)
```

File: client_server.idl

```
/* $RCSfile: client_server.idl,v $
** $Author: tommieb $
** $Date: 2002/11/08 19:40:03 $
** $Id: client_server.idl,v 1.3 2002/11/08 19:40:03 tommieb Exp $
*/
[ uuid(98e4bfa6-f323-11d6-823e-000000000000), version(1.1), pointer_default(ptr) ]

interface tommieb
{
      typedef [ptr, string] char * string_t;

      typedef struct
      {
            unsigned32 argc;
            [size_is(argc)] string_t argv[];
      }args;

      /* My interface definition goes here */
      boolean AddIt(
            [in]                  handle_t            h,
            [in]                  long                  in_num1,
            [in]                  long                  in_num2,
            [out, ref]         long                 *total,
            [out, ref]         error_status_t   *status
        );
      boolean Stringize(
            [in]               handle_t            h,
            [in]               args                  *in_str,
            [out]              args                  sz_str,
            [out, ref]      error_status_t      *status
```

```
      );
      boolean Compress(
            [in]                    handle_t            h,
            [in]                    args                  *orig_data,
            [out]                   args                  **compressed_data,
            [out, ref]          error_status_t   *status
      );
}
```

File: mc.h (Generated by IDL compiler - shown here for completion)

```
/* Generated by IDL compiler version OSF DCE T1.1.0-03 with GNU Flex/Bison */
#ifndef tommieb_v1_1_included
#define tommieb_v1_1_included
#ifndef IDLBASE_H
#include <dce/idlbase.h>
#endif
#include <dce/rpc.h>

#ifdef __cplusplus
    extern "C" {
#endif

#ifndef nbase_v0_0_included
#include <dce/nbase.h>
#endif
typedef idl_char *string_t;
typedef struct  {
unsigned32 argc;
string_t argv[1];
} args;
extern idl_boolean AddIt(
#ifdef IDL_PROTOTYPES
    /* [in] */ handle_t h,
    /* [in] */ idl_long_int in_num1,
    /* [in] */ idl_long_int in_num2,
    /* [out] */ idl_long_int *total,
    /* [out] */ error_status_t *status
#endif
);
extern idl_boolean Stringize(
#ifdef IDL_PROTOTYPES
    /* [in] */ handle_t h,
    /* [in] */ args *in_str,
    /* [out] */ args **sz_str,
    /* [out] */ error_status_t *status
#endif
);
extern idl_boolean Compress(
#ifdef IDL_PROTOTYPES
    /* [in] */ handle_t h,
    /* [in] */ args *orig_data,
    /* [out] */ args **compressed_data,
    /* [out] */ error_status_t *status
#endif
);
typedef struct tommieb_v1_1_epv_t {
idl_boolean (*AddIt)(
#ifdef IDL_PROTOTYPES
    /* [in] */ handle_t h,
    /* [in] */ idl_long_int in_num1,
    /* [in] */ idl_long_int in_num2,
    /* [out] */ idl_long_int *total,
    /* [out] */ error_status_t *status
#endif
);
idl_boolean (*Stringize)(
#ifdef IDL_PROTOTYPES
    /* [in] */ handle_t h,
    /* [in] */ args *in_str,
    /* [out] */ args **sz_str,
    /* [out] */ error_status_t *status
#endif
);
idl_boolean (*Compress)(
#ifdef IDL_PROTOTYPES
```

```
    /* [in] */ handle_t h,
    /* [in] */ args *orig_data,
    /* [out] */ args **compressed_data,
    /* [out] */ error_status_t *status
#endif
);
} tommieb_v1_1_epv_t;
extern rpc_if_handle_t tommieb_v1_1_c_ifspec;
extern rpc_if_handle_t tommieb_v1_1_s_ifspec;

#ifdef __cplusplus
    }
#endif

#endif
```

File: dce_misc.h

```
/*
** File: "$RCSfile: dce_misc.h,v $"
** Author: "$Author: tommieb $"
** Date: "$Date: 2002/11/08 15:18:23 $"
** Purpose: Header file for miscellany DCE routines
*/
#include <dce/dce_error.h>

#ifndef __DCE_MISC_H
#define __DCE_MISC_H

#ifdef __cplusplus
extern "C" {
#endif

static char *RCSIdHdr1 __attribute__((__unused__)) = "$Id: dce_misc.h,v 1.2 2002/11/08
15:18:23 tommieb Exp $";

/* dce_misc.c */
void chk_dce_err(error_status_t, const char *, const char *, unsigned int);
int get_client_rpc_binding(rpc_binding_handle_t *, char *, rpc_if_handle_t, char *);

#ifdef __cplusplus
}
#endif

#endif /* __DCE_MISC_H */
```

File: dce_misc.c

```
/*
** File: "$RCSfile: dce_misc.c,v $"
** Author: "$Author: tommieb $"
** Date: "$Date: 2002/11/08 13:09:41 $"
** Purpose: General DCE routines
*/

#include "main_client.h"

static char *Id __attribute__((__unused__)) = "$Id: dce_misc.c,v 1.3 2002/11/08
13:09:41 tommieb Exp $";

void chk_dce_err(error_status_t ecode, const char *where, const char *why, unsigned
int fatal)
{
      dce_error_string_t errstr;
      int error_status;
      if (ecode != error_status_ok)
      {
            dce_error_inq_text(ecode, errstr, &error_status);
            /* if (error_status == error_status_ok) */
            printf("\n>>>> DCE Error <<<<\nLast exec'd DCE stmt: %s\nFunction:
%s\nError code: 0x%lx\nReason: %s\n", where, why, ecode, errstr);
            /* else
            printf("ERROR.  where = <%s> why = <%s> error code = 0x%lx\n", where, why,
ecode); */
            if (fatal)
                  exit(EXIT_FAILURE);
```

```
    }
}

int get_client_rpc_binding(rpc_binding_handle_t *binding_handle, char *hostname,
rpc_if_handle_t interface_spec,char *protocol)
{
      char * resolved_binding;
      char * printable_uuid __attribute__((__unused__));
      char * protocol_family;
      char partial_string_binding[128];
      rpc_if_id_t interface __attribute__((__unused__));
      uuid_t ifc_uuid __attribute__((__unused__));
      error_status_t status;

      /*
       * create a string binding given the command line parameters and
       * resolve it into a full binding handle using the endpoint mapper.
       *  The binding handle resolution is handled by the runtime library
       */
      if (strcmp(protocol, "udp"))
            protocol_family = "ncadg_ip_udp";
      else
            protocol_family = "ncacn_ip_tcp";
      sprintf(partial_string_binding, "%s:%s[]", protocol_family, hostname);
      rpc_binding_from_string_binding((unsigned char *)partial_string_binding,
binding_handle, &status);
      chk_dce_err(status, "rpc_binding_from_string_binding(...)",
"get_client_rpc_binding(...)", 1);
      /*
       * Resolve the partial binding handle using the endpoint mapper
       */
      rpc_ep_resolve_binding(*binding_handle, interface_spec, &status);
      chk_dce_err(status, "rpc_ep_resolve_binding(...)",
"get_client_rpc_binding(...)", 1);
      /*
       * Get a printable rendition of the binding handle and echo to
       * the user.
       */
      rpc_binding_to_string_binding(*binding_handle, (unsigned char
**)&resolved_binding, &status);
      chk_dce_err(status, "rpc_binding_to_string_binding(...)",
"get_client_rpc_binding(...)", 1);
      printf("fully resolving binding for server is: %s\n", resolved_binding);
      return 1;
}
```

File: main_client.h

```
/*
** File: "$RCSfile: main_client.h,v $"
** Author: "$Author: tommieb $"
** Date: "$Date: 2002/11/09 17:23:37 $"
** Purpose: Header file for client side.
*/

#ifndef __MAIN_CLIENT_H
#define __MAIN_CLIENT_H

#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>
#include <signal.h>
#include <getopt.h>
#include <unistd.h>
#include <curses.h>
#include <dce/rpc.h>
#include <dce/dce_error.h>
#include <dce/pthread_exc.h>

#ifdef __cplusplus
extern "C" {
#endif

static char *RCSIdHdr __attribute__((__unused__)) = "$Id: main_client.h,v 1.7
2002/11/09 17:23:37 tommieb Exp $";
```

```
extern unsigned32 status;
extern rpc_binding_handle_t impl_server;
extern int verbose;

/* main_client.c */
void cleanup(void);
void panic(const char *, ...);
void init_signals(void);

/* menu_client.c */
int draw_menu(void);
int process_two_nums(void);
void process_string(int);

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_CLIENT_H */
```

File: menu_client.c

```
/*
** File: "$RCSfile: menu_client.c,v $"
** Author: "$Author: tommieb $"
** Date: "$Date: 2002/11/09 17:40:28 $"
** Purpose: This is where the menu options are processed...
*/

#include "main_client.h"
#include "dce_misc.h"
#include "mc.h"

static char *Id __attribute__((__unused__)) = "$Id: menu_client.c,v 1.10 2002/11/09
17:40:28 tommieb Exp $";

int draw_menu(void)
{
      char str_num[2];
      int num, valid_num = 0;
      mvprintw(3, 2, "----> Client <----");
      mvprintw(5, 2, "1. Add two numbers.\n\n");
      mvprintw(7, 2, "2. Input a string.\n\n");
      mvprintw(9, 2, "3. Compress a string.\n\n");
      mvprintw(11, 2, "9. Goodbye!\n");
      refresh();
      crmode();
      do
      {
            mvprintw(13, 2, "Enter Choice (9 to Quit): ");
            refresh();
            clrtoeol();
            getnstr(str_num, 2);
            sscanf(str_num, "%d", &num);
            if ((num >= 1) && (num <= 9))
                  valid_num = 1;
      }while (!valid_num);
      return num;
}

int process_two_nums(void)
{
      char str_num[2];
      int valid_num = 0, quit = 0, rv;
      long num1, num2, Total;
      clear();
      mvprintw(3, 2, "----> Client. (1. Add two numbers) <----");
      refresh();
      nocrmode();
      do
      {
            mvprintw(5, 2, "Enter first number (0 to exit): ");
            refresh();
            clrtoeol();
            getnstr(str_num, 2);
            rv = sscanf(str_num, "%ld", &num1);
```

```
            if (num1 == 0 || rv == 0)
                  quit = 1;
            else
                  valid_num = 1;
      }while (!valid_num);
      if (quit)
      {
            mvprintw(7, 2, "User quitted!");
            crmode();
            getch();
            return 1;
      }
      valid_num = 0;
      nocrmode();
      do
      {
            mvprintw(7, 2, "Enter second number (0 to exit): ");
            getnstr(str_num, 2);
            clrtoeol();
            rv = sscanf(str_num, "%ld", &num2);
            if (num2 == 0 || rv == 0)
                  quit = 1;
            else
                  valid_num = 1;
      }while (!valid_num);
      if (quit)
      {
            crmode();
            mvprintw(9, 2, "User quitted!");
            getch();
            return 1;
      }
      if (verbose)
            /* Call the server! */
            mvprintw(9, 2, "Calling the server....");
      rv = AddIt(impl_server, num1, num2, &Total, &status);
      if (rv && status == error_status_ok)
            if (verbose)
                  mvprintw(11, 2, "Got result from server....%ld", Total);
            else
                  mvprintw(11, 2, "Result is %ld", Total);
      if (status != error_status_ok)
            chk_dce_err(status, "Addit(...)", "process_two_nums(...)", 1);
      refresh();
      crmode();
      mvprintw(13, 25, "Press a key...");
      getch();
      return 0;
}

void process_string(int compress)
{
#define MAX_INPUT      30
      char str_inp[MAX_INPUT];
      args *in_args, *out_args;
      long rv;
      clear();
      nocrmode();
      if (!compress)
            mvprintw(3, 2, "----> Client. (2. Input a string) <----");
      else
            mvprintw(3, 2, "----> Client. (3. Compress a string) <----");
      mvprintw(5, 2, "Enter a string (No entry to exit): ");
      in_args = (args *) malloc(sizeof(args) + (MAX_INPUT * sizeof(string_t)));
      if (in_args == NULL)
            return;
      getnstr(str_inp, sizeof(str_inp));
      if (!strlen(str_inp))
      {
            crmode();
            mvprintw(7, 2, "User quitted!");
            getch();
            return;
      }
      in_args->argv[0] = (string_t) strdup(str_inp);
      in_args->argc = 1;
      if (verbose)
```

```
            mvprintw(7, 2, "Calling the server....");
      if (!compress)
            rv = Stringize(impl_server, in_args, &out_args, &status);
      else
            rv = Compress(impl_server, in_args, &out_args, &status);
      if (rv && status == error_status_ok)
      {
            if (compress)
            {
                  mvprintw(9, 2, "Original string was %s and length was %d", in_args-
>argv[0], strlen(in_args->argv[0]));
                  if (verbose)
                        mvprintw(11, 2, "Got result from server....");
                  mvprintw(12, 14, "Compressed string is %s and length is %d",
out_args->argv[0], strlen(out_args->argv[0]));
            }else
                  if (verbose)
                        mvprintw(9, 2, "Got result from server....%s", out_args-
>argv[0]);
                  else
                        mvprintw(9, 2, "Result is %s", out_args->argv[0]);
      }
      if (status != error_status_ok)
            chk_dce_err(status, (compress) ? "Compress(...)" : "Stringize(...)",
"process_string(...)", 1);
      refresh();
      crmode();
      mvprintw(14, 25, "Press a key...");
      getch();
      return;
}
```

File: main_client.c

```
/*
** File: "$RCSfile: main_client.c,v $"
** Author: "$Author: tommieb $"
** Date: "$Date: 2002/11/09 17:23:37 $"
** Purpose: Demonstrates a menu driven interface!
*/

#include "main_client.h"
#include "dce_misc.h"
#include "mc.h"

/* RCS Ident. */
static char *RCSId __attribute__((__unused__)) = "$Id: main_client.c,v 1.7 2002/11/09
17:23:37 tommieb Exp $";

/* Global variables pertaining to command line options */
int use_udp = 0, use_tcp = 0, verbose = 0, curses_init = 0;
char rpc_host[128] = "localhost"; /* default! */
char *protocol;

/* Signal handler struct */
struct sigaction sigact;

/* DCE handle to server! */
unsigned32 status;
rpc_binding_handle_t  impl_server;
/* Functions */
static void signal_handler(int);
static void shutdown_client(void);
static void usage(void);

int main(int argc, char **argv)
{
      extern char *optarg;
      extern int optind, opterr, optopt;
      int c, menu_done = 1, menu_opt;
      /* Process command line! */
      while ((c = getopt(argc, argv, "h:utv")) != EOF)
      {
            switch (c)
            {
                  case 'u'      :     use_udp = 1;
```

```
                                        break;
                case 't'      :     use_tcp = 1;
                                        break;
                case 'v'      :     verbose = 1;
                                        break;
                case 'h'      :     strncpy(rpc_host, optarg, sizeof(rpc_host)-1);
                                        break;
                default       :      usage();
            }
      }
      if (!use_tcp && !use_udp)
            use_tcp = 1;
      if (use_udp)
            protocol = "udp";
      else
            protocol = "tcp";
      /* Initialise curses */
      initscr();
      curses_init = 1;
      /* Set kbd to cooked mode */
      nocbreak();
      /* Set up signal handler! */
      init_signals();
      /* At first sign of trouble, we cleanup! */
      atexit(cleanup);
      /* Get binding handle to server! */
      if (get_client_rpc_binding(&impl_server, rpc_host, tommieb_v1_1_c_ifspec,
protocol) == 0)
      {
            endwin();
            curses_init = 0;
            panic("Could not obtain RPC Server binding handle\n");
      }
      /* Do the biz! */
      while (menu_done)
      {
            clear();
            menu_opt = draw_menu();
            switch(menu_opt)
            {
                  case 1      :      process_two_nums();
                                        break;
                  case 2      :      process_string(0);
                                        break;
                  case 3      :      process_string(1);
                                        break;
                  case 9      :      clear();
                                        endwin();
                                        curses_init = 0;
                                        menu_done = 0;
                                        break;
                  default      :    break;
            }
      }
      exit(EXIT_SUCCESS);
}

void panic(const char *fmt, ...)
{
      va_list ap;
      if (curses_init)
      {
            endwin();
            curses_init = 0;
      }
      va_start(ap, fmt);
      vfprintf(stderr, fmt, ap);
      va_end(ap);
      exit(EXIT_FAILURE);
}

void init_signals(void)
{
      sigact.sa_handler = signal_handler;
      sigemptyset(&sigact.sa_mask);
      sigact.sa_flags = 0;
      /* Did we get ctrl-c'd */
```

```
      sigaction(SIGINT, &sigact, (struct sigaction *) NULL);
      /* Did we bomb? */
      sigaddset(&sigact.sa_mask, SIGSEGV);
      sigaction(SIGSEGV, &sigact, (struct sigaction *) NULL);
}

static void signal_handler(int sig)
{
      if (sig == SIGINT)
            panic("Whoops! We got ^c'd....\n");
      if (sig == SIGSEGV)
            panic("Whoops! We blew up....\n");
}

void cleanup(void)
{
      /* Reset Signal handlers */
      sigact.sa_flags = SA_RESETHAND;
      sigaction(SIGINT, &sigact, 0);
      sigaction(SIGSEGV, &sigact, 0);
      /* Shutdown client gracefully! */
      if (curses_init)
      {
            endwin();
            curses_init = 0;
      }
      shutdown_client();
}

static void shutdown_client(void)
{
      rpc_binding_free(&impl_server, &status);
      chk_dce_err(status, "rpc_free_binding(...)", "shutdown_client(...)", 1);
}

static void usage(void)
{
      printf("usage:  client [-h hostname] [-u] [-t]\n");
      printf("        -h : specify host where DCE server lives\n");
      printf("        -u : use UDP protocol\n");
      printf("        -t : use TCP protocol\n");
      printf("        -v : verbose mode\n");
      exit(1);
}
```

File: main_server.c

```
/*
** File: "$RCSfile: main_server.c,v $"
** Author: "$Author: tommieb $"
** Date: "$Date: 2002/11/08 19:41:55 $"
** Purpose: The server code!
*/

#include <stdio.h>
#include <signal.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>
#include <dce/rpc.h>
#include <dce/pthread_exc.h>
#include "mc.h"
#include "dce_misc.h"

/* RCS Ident. */
static char *RCSId __attribute__((__unused__)) = "$Id: main_server.c,v 1.7 2002/11/08
19:41:55 tommieb Exp $";

/* Functions */
pthread_t sig_handler_thread;
static void signal_handler(void *);
static void wait_for_signals(void);
int rle_compress(const char *, char **, int);

int main(int argc __attribute__((__unused__)), char **argv
__attribute__((__unused__)))
```

```
{
      char *server_binding_str;
      unsigned32 status = 0,i;
      rpc_binding_vector_p_t  impl_server_bind;
      printf("Registering server....");
      rpc_server_register_if(tommieb_v1_1_s_ifspec, NULL, NULL, &status);
      chk_dce_err(status, "rpc_server_register_if(...)", "main(...)", 1);
      printf("Registered!\n");
      printf("Preparing binding handle....");
      rpc_server_use_all_protseqs(rpc_c_protseq_max_calls_default, &status);
      chk_dce_err(status, "rpc_server_use_all_protseqs(...)", "main(...)", 1);
      rpc_server_inq_bindings(&impl_server_bind, &status);
      chk_dce_err(status, "rpc_server_inq_bindings(...)", "main(...)", 1);
      printf("Prepared!\n");
      printf("Registering bindings with endpoint mapper....");
      rpc_ep_register(tommieb_v1_1_s_ifspec, impl_server_bind, NULL, (unsigned char
*)"tommieb_server", &status);
      chk_dce_err(status, "rpc_ep_register(...)", "main(...)", 1);
      printf("Registered!\n");
      printf("Server's communication endpoints are...\n");
      for (i = 0; i < impl_server_bind->count; i++)
      {
            rpc_binding_to_string_binding(impl_server_bind->binding_h[i], (unsigned
char **)&server_binding_str, &status);
            if (server_binding_str)
                  printf("\t%s\n", server_binding_str);
      }
      wait_for_signals();
      printf("Listening for calls....\n");
      TRY
      {
            rpc_server_listen(rpc_c_listen_max_calls_default, &status);
      }
      CATCH_ALL
      {
            printf("Server stopped....\n");
      }
      ENDTRY
      printf("Killing the server's signal handler thread....\n");
      pthread_cancel(sig_handler_thread);
      printf("Unregistering bindings from endpoint mapper....");
      rpc_ep_unregister(tommieb_v1_1_s_ifspec, impl_server_bind, NULL, &status);
      chk_dce_err(status, "rpc_ep_unregister(...)", "main(...)", 1);
      printf("Unregistered!\n");
      printf("Cleaning up communications endpoint....");
      rpc_server_unregister_if(tommieb_v1_1_s_ifspec, NULL, &status);
      chk_dce_err(status, "rpc_server_unregister_if(...)", "main(...)", 1);
      printf("Cleaned up!\n");
      exit(EXIT_SUCCESS);
}

static void wait_for_signals(void)
{
      sigset_t default_signal_mask;
      sigset_t old_signal_mask;
      sigemptyset(&default_signal_mask);
      pthread_sigmask(SIG_BLOCK, &default_signal_mask, &old_signal_mask);
      pthread_create(&sig_handler_thread, pthread_attr_default, (void
*)signal_handler, NULL);
}

static void signal_handler(void *arg __attribute__((__unused__)))
{
      sigset_t catch_signal_mask;
      sigset_t old_signal_mask;
      int which_signal;
      unsigned32 status;
      sigemptyset(&catch_signal_mask);
      sigaddset(&catch_signal_mask, SIGINT);
      pthread_sigmask(SIG_BLOCK, &catch_signal_mask, &old_signal_mask);
      while (1)
      {
            sigwait(&catch_signal_mask, &which_signal);
            if ((which_signal == SIGINT) ||
                  (which_signal == SIGQUIT) ||
                  (which_signal == SIGTERM) ||
                  (which_signal == SIGKILL) ||
```

```
                  (which_signal == SIGHUP))
                  rpc_mgmt_stop_server_listening(NULL, &status);
      }
}

idl_boolean AddIt(rpc_binding_handle_t h, long in_num1, long in_num2, long *Total,
error_status_t *status)
{
      char *binding_info;
      error_status_t e;
      rpc_binding_to_string_binding(h, (unsigned char **)&binding_info, &e);
      if (e == rpc_s_ok)
            printf("AddIt(...) called by client: %s\n", binding_info);
      printf("AddIt's parameters are: %ld, %ld\n", in_num1, in_num2);
      *Total = in_num1 + in_num2;
      *status = error_status_ok;
      return 1;
}

idl_boolean Stringize(rpc_binding_handle_t h, args *in_args, args **out_args,
error_status_t *status)
{
      char *binding_info, *ptr, *tmp_ptr;
      error_status_t e;
      unsigned tmp_len;
      args *tmp;
      rpc_binding_to_string_binding(h, (unsigned char **)&binding_info, &e);
      if (e == rpc_s_ok)
            printf("Stringize(...) called by client: %s\n", binding_info);
      printf("Stringize's parameters are: %s\n", in_args->argv[0]);
      ptr = in_args->argv[0];
      tmp_len = sizeof(args) + (in_args->argc * sizeof(string_t *));
      tmp = (args *) rpc_ss_allocate(tmp_len);
      tmp->argc = in_args->argc;
      tmp->argv[0] = (string_t) rpc_ss_allocate(strlen(in_args->argv[0]) + 1);
      tmp_ptr = tmp->argv[0];
      while (*ptr)
      {
            if (isupper(*ptr) && isalpha(*ptr))
                  *tmp_ptr = tolower(*ptr);
            else
                  if (islower(*ptr) && isalpha(*ptr))
                        *tmp_ptr = toupper(*ptr);
                  else
                        *tmp_ptr = *ptr;
            *tmp_ptr++;
            *ptr++;
      }
      *tmp_ptr = '\0';
      *out_args = tmp;
      *status = error_status_ok;
      return 1;
}

idl_boolean Compress(rpc_binding_handle_t h, args *orig_data, args **compressed_data,
error_status_t *status)
{
      error_status_t e;
      char *binding_info;
      args *tmp;
      unsigned tmp_len;
      rpc_binding_to_string_binding(h, (unsigned char **)&binding_info, &e);
      if (e == rpc_s_ok)
            printf("Compress(...) called by client: %s\n", binding_info);
      printf("Compress's parameters are: %s\n", orig_data->argv[0]);
      tmp_len = sizeof(args) + (orig_data->argc * sizeof(string_t *));
      tmp = (args *) rpc_ss_allocate(tmp_len);
      tmp->argc = orig_data->argc;
      if (rle_compress(orig_data->argv[0], &tmp->argv[0], strlen(orig_data->argv[0]) +
1) == 2)
            tmp->argv[0] = NULL;
       /* rle_compress same code as in for sockets in Appendix B!!!! */
      *compressed_data = tmp;
      *status = error_status_ok;
}
```

Appendix D - Original posting to usenet on the birth of Linux.

To: Linux-Activists@BLOOM-PICAYUNE.MIT.EDU
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Subject: Birthday (was Re: Uptime found. Thanks to all)
Date: 31 Jul 92 22:15:20 GMT

In article <1992Jul30.211132.20101@cc.umontreal.ca> duperval@ERE.UMontreal.CA (Duperval Laurent) writes:
>
>P.S. BTW, noone answered yet: when is Linux's birthday? Let's have a
>party!

I couldn't for the life of me remember when it all happened, and I don't keep a diary, so I can't give you any exact dates for when linux "was born". But I did start to wonder, so I started ftp'ing around for archives of the comp.os.minix group (where I announced it), and this is what I came up with (with some editing).

This is just a sentimental journey into some of the first posts concerning linux, so you can happily press 'n' now if you actually thought you'd get anything technical.

> From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
> Newsgroups: comp.os.minix
> Subject: Gcc-1.40 and a posix-question
> Message-ID: <1991Jul3.100050.9886@klaava.Helsinki.FI>
> Date: 3 Jul 91 10:00:50 GMT
>
> Hello netlanders,
>
> Due to a project I'm working on (in minix), I'm interested in the posix
> standard definition. Could somebody please point me to a (preferably)
> machine-readable format of the latest posix rules? Ftp-sites would be
> nice.

The project was obviously linux, so by July 3rd I had started to think about actual user-level things: some of the device drivers were ready, and the harddisk actually worked. Not too much else.

> As an aside for all using gcc on minix - [deleted]

Just a success-report on porting gcc-1.40 to minix using the 1.37 version made by Alan W Black & co.

> Linus Torvalds torvalds@kruuna.helsinki.fi
>
> PS. Could someone please try to finger me from overseas, as I've
> installed a "changing .plan" (made by your's truly), and I'm not certain
> it works from outside? It should report a new .plan every time.

So I was clueless - had just learned about named pipes. Sue me. This part of the post got a lot more response than the actual POSIX query, but the query did lure out arl from the woodwork, and we mailed around for a bit, resulting in the Linux subdirectory on nic.funet.fi.

Then, almost two months later, I actually had something working: I made sources for version 0.01 available on nic sometimes around this time. 0.01 sources weren't actually runnable: they were just a token gesture to arl who had probably started to despair about ever getting anything. This next post must have been from just a couple of weeks before that release.

> From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
> Newsgroups: comp.os.minix
> Subject: What would you like to see most in minix?
> Summary: small poll for my new operating system
> Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
> Date: 25 Aug 91 20:57:08 GMT
> Organization: University of Helsinki
>
>
> Hello everybody out there using minix -
>
> I'm doing a (free) operating system (just a hobby, won't be big and

> professional like gnu) for 386(486) AT clones. This has been brewing
> since april, and is starting to get ready. I'd like any feedback on
> things people like/dislike in minix, as my OS resembles it somewhat
> (same physical layout of the file-system (due to practical reasons)
> among other things).
>
> I've currently ported bash(1.08) and gcc(1.40), and things seem to work.
> This implies that I'll get something practical within a few months, and
> I'd like to know what features most people would want. Any suggestions
> are welcome, but I won't promise I'll implement them :-)
>
> Linus (torvalds@kruuna.helsinki.fi)
>
> PS. Yes - it's free of any minix code, and it has a multi-threaded fs.
> It is NOT protable (uses 386 task switching etc), and it probably never
> will support anything other than AT-harddisks, as that's all I have :-(.

Judging from the post, 0.01 wasn't actually out yet, but it's close. I'd guess the first version went out in the middle of September -91. I got some responses to this (most by mail, which I haven't saved), and I even got a few mails asking to be beta-testers for linux.

After that just a few general answers to quesions on the net:

> From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
> Newsgroups: comp.os.minix
> Subject: Re: What would you like to see most in minix?
> Summary: yes - it's nonportable
> Message-ID: <1991Aug26.110602.19446@klaava.Helsinki.FI>
> Date: 26 Aug 91 11:06:02 GMT
> Organization: University of Helsinki
>
> In article <1991Aug25.234450.22562@nntp.hut.fi> jkp@cs.HUT.FI (Jyrki Kuoppala) writes:
> >> [re: my post about my new OS]
> > >
> > >Tell us more! Does it need a MMU?
> >
> > Yes, it needs a MMU (sorry everybody), and it specifically needs a
> > 386/486 MMU (see later).
> >
> > >
> > >>PS. Yes - it's free of any minix code, and it has a multi-threaded fs.
> > >>It is NOT protable (uses 386 task switching etc)
> > >
> > >How much of it is in C? What difficulties will there be in porting?
> > >Nobody will believe you about non-portability ;-), and I for one would
> > >like to port it to my Amiga (Mach needs a MMU and Minix is not free).
> >
> > Simply, I'd say that porting is impossible. It's mostly in C, but most
> > people wouldn't call what I write C. It uses every conceivable feature
> > of the 386 I could find, as it was also a project to teach me about the
> > 386. As already mentioned, it uses a MMU, for both paging (not to disk
> > yet) and segmentation. It's the segmentation that makes it REALLY 386
> > dependent (every task has a 64Mb segment for code & data - max 64 tasks
> > in 4Gb. Anybody who needs more than 64Mb/task - tough cookies).
> >
> > It also uses every feature of gcc I could find, specifically the __asm__
> > directive, so that I wouldn't need so much assembly language objects.
> > Some of my "C"-files (specifically mm.c) are almost as much assembler as
> > C. It would be "interesting" even to port it to another compiler (though
> > why anybody would want to use anything other than gcc is a mystery).

[editors note: linux has in fact gotten more portable with newer versions: there was a lot more assembly in the early versions. Not that anybody in their right mind would try to port it even now]

> > Unlike minix, I also happen to LIKE interrupts, so interrupts are
> > handled without trying to hide the reason behind them (I especially like
> > my hard-disk-driver. Anybody else make interrupts drive a state-
> > machine?). All in all it's a porters nightmare.
> >
> > >As for the features; well, pseudo ttys, BSD sockets, user-mode
> > >filesystems (so I can say cat /dev/tcp/kruuna.helsinki.fi/finger),
> > >window size in the tty structure, system calls capable of supporting
> > >POSIX.1. Oh, and bsd-style long file names.

>
> Most of these seem possible (the tty structure already has stubs for
> window size), except maybe for the user-mode filesystems. As to POSIX,
> I'd be delighted to have it, but posix wants money for their papers, so
> that's not currently an option. In any case these are things that won't
> be supported for some time yet (first I'll make it a simple minix-
> lookalike, keyword SIMPLE).
>
> Linus (torvalds@kruuna.helsinki.fi)
>
> PS. To make things really clear - yes I can run gcc on it, and bash, and
> most of the gnu [bin/file]utilities, but it's not very debugged, and the
> library is really minimal. It doesn't even support floppy-disks yet. It
> won't be ready for distribution for a couple of months. Even then it
> probably won't be able to do much more than minix, and much less in some
> respects. It will be free though (probably under gnu-license or similar).

Well, obviously something worked on my machine: I doubt I had yet gotten
gcc to compile itself under linux (or I would have been too proud of it
not to mention it). Still before any release-date.

Then, October 5th, I seem to have released 0.02. As I already
mentioned, 0.01 didn't actually come with any binaries: it was just
source code for people interested in what linux looked like. Note the
lack of announcement for 0.01: I wasn't too proud of it, so I think I
only sent a note to everybody who had shown interest.

> From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
> Newsgroups: comp.os.minix
> Subject: Free minix-like kernel sources for 386-AT
> Message-ID: <1991Oct5.054106.4647@klaava.Helsinki.FI>
> Date: 5 Oct 91 05:41:06 GMT
> Organization: University of Helsinki
>
> Do you pine for the nice days of minix-1.1, when men were men and wrote
> their own device drivers? Are you without a nice project and just dying
> to cut your teeth on a OS you can try to modify for your needs? Are you
> finding it frustrating when everything works on minix? No more all-
> nighters to get a nifty program working? Then this post might be just
> for you :-)
>
> As I mentioned a month(?) ago, I'm working on a free version of a
> minix-lookalike for AT-386 computers. It has finally reached the stage
> where it's even usable (though may not be depending on what you want),
> and I am willing to put out the sources for wider distribution. It is
> just version 0.02 (+1 (very small) patch already), but I've successfully
> run bash/gcc/gnu-make/gnu-sed/compress etc under it.
>
> Sources for this pet project of mine can be found at nic.funet.fi
> (128.214.6.100) in the directory /pub/OS/Linux. The directory also
> contains some README-file and a couple of binaries to work under linux
> (bash, update and gcc, what more can you ask for :-). Full kernel
> source is provided, as no minix code has been used. Library sources are
> only partially free, so that cannot be distributed currently. The
> system is able to compile "as-is" and has been known to work. Heh.
> Sources to the binaries (bash and gcc) can be found at the same place in
> /pub/gnu.
>
> ALERT! WARNING! NOTE! These sources still need minix-386 to be compiled
> (and gcc-1.40, possibly 1.37.1, haven't tested), and you need minix to
> set it up if you want to run it, so it is not yet a standalone system
> for those of you without minix. I'm working on it. You also need to be
> something of a hacker to set it up (?), so for those hoping for an
> alternative to minix-386, please ignore me. It is currently meant for
> hackers interested in operating systems and 386's with access to minix.
>
> The system needs an AT-compatible harddisk (IDE is fine) and EGA/VGA. If
> you are still interested, please ftp the README/RELNOTES, and/or mail me
> for additional info.
>
> I can (well, almost) hear you asking yourselves "why?". Hurd will be
> out in a year (or two, or next month, who knows), and I've already got
> minix. This is a program for hackers by a hacker. I've enjouyed doing
> it, and somebody might enjoy looking at it and even modifying it for
> their own needs. It is still small enough to understand, use and

> modify, and I'm looking forward to any comments you might have.
>
> I'm also interested in hearing from anybody who has written any of the
> utilities/library functions for minix. If your efforts are freely
> distributable (under copyright or even public domain), I'd like to hear
> from you, so I can add them to the system. I'm using Earl Chews estdio
> right now (thanks for a nice and working system Earl), and similar works
> will be very wellcome. Your (C)'s will of course be left intact. Drop me
> a line if you are willing to let me use your code.
>
> Linus
>
> PS. to PHIL NELSON! I'm unable to get through to you, and keep getting
> "forward error - strawberry unknown domain" or something.

Well, it doesn't sound like much of a system, does it? It did work, and some people even tried it out. There were several bad bugs (and there was no floppy-driver, no VM, no nothing), and 0.02 wasn't really very useable.

0.03 got released shortly thereafter (max 2-3 weeks was the time between releases even back then), and 0.03 was pretty useable. The next version was numbered 0.10, as things actually started to work pretty well. The next post gives some idea of what had happened in two months more...

> From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
> Newsgroups: comp.os.minix
> Subject: Re: Status of LINUX?
> Summary: Still in beta
> Message-ID: <1991Dec19.233545.8114@klaava.Helsinki.FI>
> Date: 19 Dec 91 23:35:45 GMT
> Organization: University of Helsinki
>
> In article <469@htsa.htsa.aha.nl> miquels@maestro.htsa.aha.nl (Miquel van Smoorenburg) writes:
> >Hello *,
> > I know some people are working on a FREE O/S for the 386/486,
> >under the name Linux. I checked nic.funet.fi now and then, to see what was
> >happening. However, for the time being I am without FTP access so I don't
> >know what is going on at the moment. Could someone please inform me about it?
> >It's maybe best to follow up to this article, as I think that there are
> >a lot of potential interested people reading this group. Note, that I don't
> >really *have* a >= 386, but I'm sure in time I will.
>
> Linux is still in beta (although available for brave souls by ftp), and
> has reached the version 0.11. It's still not as comprehensive as
> 386-minix, but better in some respects. The "Linux info-sheet" should
> be posted here some day by the person that keeps that up to date. In
> the meantime, I'll give some small pointers.
>
> First the bad news:
>
> - Still no SCSI: people are working on that, but no date yet.
> Thus you need a AT-interface disk (I have one report that it
> works on an EISA 486 with a SCSI disk that emulates the
> AT-interface, but that's more of a fluke than anything else:
> ISA+AT-disk is currently the hardware setup)

As you can see, 0.11 had already a small following. It wasn't much, but it did work.

> - still no init/login: you get into bash as root upon bootup.

That was still standard in the next release.

> - although I have a somewhat working VM (paging to disk), it's not
> ready yet. Thus linux needs at least 4M to be able to run the
> GNU binaries (especially gcc). It boots up in 2M, but you
> cannot compile.

I actually released a 0.11+VM version just before Christmas -91: I didn't need it myself, but people were trying to compile the kernel in 2MB and failing, so I had to implement it. The 0.11+VM version was available only to a small number of people that wanted to test it out: I'm still surprised it worked as well as it did.

> - minix still has a lot more users: better support.
>
> - it hasn't got years of testing by thousands of people, so there
> are probably quite a few bugs yet.
>
> Then for the good things..
>
> - It's free (copyright by me, but freely distributable under a
> very lenient copyright)

The early copyright was in fact much more restrictive than the GNU
copyleft: I didn't allow any money at all to change hands due to linux.
That changed with 0.12.

> - it's fun to hack on.
>
> - /real/ multithreading filesystem.
>
> - uses the 386-features. Thus locked into the 386/486 family, but
> it makes things clearer when you don't have to cater to other
> chips.
>
> - a lot more... read my .plan.
>
> /I/ think it's better than minix, but I'm a bit prejudiced. It will
> never be the kind of professional OS that Hurd will be (in the next
> century or so :), but it's a nice learning tool (even more so than
> minix, IMHO), and it was/is fun working on it.
>
> Linus (torvalds@kruuna.helsinki.fi)
>
> ---- my .plan -------------------------
> Free UNIX for the 386 - coming 4QR 91 or 1QR 92.
>
> The current version of linux is 0.11 - it has most things a unix kernel
> needs, and will probably be released as 1.0 as soon as it gets a little
> more testing, and we can get a init/login going. Currently you get
> dumped into a shell as root upon bootup.
>
> Linux can be gotten by anonymous ftp from 'nic.funet.fi' (128.214.6.100)
> in the directory '/pub/OS/Linux'. The same directory also contains some
> binary files to run under Linux. Currently gcc, bash, update, uemacs,
> tar, make and fileutils. Several people have gotten a running system,
> but it's still a hackers kernel.
>
> Linux still requires a AT-compatible disk to be useful: people are
> working on a SCSI-driver, but I don't know when it will be ready.
>
> There are now a couple of other sites containing linux, as people have
> had difficulties with connecting to nic. The sites are:
> Tupac-Amaru.Informatik.RWTH-Aachen.DE (137.226.112.31):
> directory /pub/msdos/replace
> tsx-11.mit.edu (18.172.1.2):
> directory /pub/linux
>
> There is also a mailing list set up 'Linux-activists@niksula.hut.fi'.
> To join, mail a request to 'Linux-activists-request@niksula.hut.fi'.
> It's no use mailing me: I have no actual contact with the mailing-list
> (other than being on it, naturally).
>
> Mail me for more info:
>
> Linus (torvalds@kruuna.Helsinki.FI)
>
> 0.11 has these new things:
>
> - demand loading
> - code/data sharing between unrelated processes
> - much better floppy drivers (they actually work mostly)
> - bug-corrections
> - support for Hercules/MDA/CGA/EGA/VGA
> - the console also beeps (WoW! Wonder-kernel :-)
> - mkfs/fsck/fdisk
> - US/German/French/Finnish keyboards
> - settable line-speeds for com1/2

As you can see: 0.11 was actually stand-alone: I wrote the first
mkfs/fsck/fdisk programs for it, so that you didn't need minix any more
to set it up. Also, serial lines had been hard-coded to 2400bps, as that
was all I had.

> Still lacking:
> - init/login
> - rename system call
> - named pipes
> - symbolic links

Well, they are all there now: init/login didn't quite make it to 0.12,
and rename() was implemented as a patch somewhere between 0.12 and 0.95.
Symlinks were in 0.95, but named pipes didn't make it until 0.96.

> 0.12 will probably be out in January (15th or so), and will have:
> - POSIX job control (by tytso)
> - VM (paging to disk)
> - Minor corrections

Actually, 0.12 was out January 5th, and contained major corrections. It
was in fact a very stable kernel: it worked on a lot of new hardware,
and there was no need for patches for a long time. 0.12 was also the
kernel that "made it": that's when linux started to spread a lot faster.
Earlier kernel releases were very much only for hackers: 0.12 actually
worked quite well.

That's all I found for 1991 - maybe it answered some questions.

Linus

To: Linux-Activists@BLOOM-PICAYUNE.MIT.EDU
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Subject: Re: Writing an OS - questions !!
Date: 5 May 92 07:58:17 GMT

In article <10685@inews.intel.com> nani@td2cad.intel.com (V. Narayanan) writes:
>
>Hi folks,
> For quite some time this "novice" has been wondering as to how one goes
>about the task of writing an OS from "scratch". So here are some questions,
>and I would appreciate if you could take time to answer 'em.

Well, I see someone else already answered, but I thought I'd take on the
linux-specific parts. Just my personal experiences, and I don't know
how normal those are.

>1) How would you typically debug the kernel during the development phase?

Depends on both the machine and how far you have gotten on the kernel:
on more simple systems it's generally easier to set up. Here's what I
had to do on a 386 in protected mode.

The worst part is starting off: after you have even a minimal system you
can use printf etc, but moving to protected mode on a 386 isn't fun,
especially if you at first don't know the architecture very well. It's
distressingly easy to reboot the system at this stage: if the 386
notices something is wrong, it shuts down and reboots - you don't even
get a chance to see what's wrong.

Printf() isn't very useful - a reboot also clears the screen, and
anyway, you have to have access to video-mem, which might fail if your
segments are incorrect etc. Don't even think about debuggers: no
debugger I know of can follow a 386 into protected mode. A 386 emulator
might do the job, or some heavy hardware, but that isn't usually
feasible.

What I used was a simple killing-loop: I put in statements like

```
die:
        jmp die
```

at strategic places. If it locked up, you were ok, if it rebooted, you knew at least it happened before the die-loop. Alternatively, you might use the sound io ports for some sound-clues, but as I had no experience with PC hardware, I didn't even use that. I'm not saying this is the only way: I didn't start off to write a kernel, I just wanted to explore the 386 task-switching primitives etc, and that's how I started off (in about April-91).
After you have a minimal system up and can use the screen for output, it gets a bit easier, but that's when you have to enable interrupts. Bang, instant reboot, and back to the old way. All in all, it took about 2 months for me to get all the 386 things pretty well sorted out so that I no longer had to count on avoiding rebooting at once, and having the basic things set up (paging, timer-interrupt and a simple task-switcher to test out the segments etc).

>2) Can you test the kernel functionality by running it as a process on a
> different OS? Wouldn't the OS(the development environment) generate
> exceptions in cases when the kernel (of the new OS) tries to modify
> 'priviledged' registers?

Yes, it's generally possible for some things, but eg device drivers usually have to be tested out on the bare machine. I used minix to develop linux, so I had no access to IO registers, interrupts etc. Under DOS it would have been possible to get access to all these, but then you don't have 32-bit mode. Intel isn't that great - it would probably have been much easier on a 68040 or similar.

So after getting a simple task-switcher (it switched between two processes that printed AAAA... and BBBB... respectively by using the timer-interrupt - Gods I was proud over that), I still had to continue debugging basically by using printf. The first thing written was the keyboard driver: that's the reason it's still written completely in assembler (I didn't dare move to C yet - I was still debugging at about instruction-level).

After that I wrote the serial drivers, and voila, I had a simple terminal program running (well, not that simple actually). It was still the same two processes (AAA..), but now they read and wrote to the console/serial lines instead. I had to reboot to get out of it all, but it was a simple kernel.

After that is was plain sailing: hairy coding still, but I had some devices, and debugging was easier. I started using C at this stage, and it certainly speeds up developement. This is also when I start to get serious about my megalomaniac ideas to make "a better minix that minix". I was hoping I'd be able to recompile gcc under linux some day...

The harddisk driver was more of the same: this time the problems with bad documentation started to crop up. The PC may be the most used architecture in the world right now, but that doesn't mean the docs are any better: in fact I haven't seen /any/ book even mentioning the weird 386-387 coupling in an AT etc (Thanks Bruce).

After that, a small filesystem, and voila, you have a minimal unix. Two months for basic setups, but then only slightly longer until I had a disk-driver (seriously buggy, but it happened to work on my machine) and a small filesystem. That was about when I made 0.01 available (late august-91? Something like that): it wasn't pretty, it had no floppy driver, and it couldn't do much anything. I don't think anybody ever compiled that version. But by then I was hooked, and didn't want to stop until I could chuck out minix.

>3) Would new linkers and loaders have to be written before you get a basic
> kernel running?

All versions up to about 0.11 were crosscompiled under minix386 - as were the user programs. I got bash and gcc eventually working under 0.02, and while a race-condition in the buffer-cache code prevented me from recompiling gcc with itself, I was able to tackle smaller compiles. 0.03 (October?) was able to recompile gcc under itself, and I think that's the first version that anybody else actually used. Still no floppies, but most of the basic things worked.

Afetr 0.03 I decided that the next version was actually useable (it was, kind of, but boy is X under 0.96 more impressive), and I called the next

version 0.10 (November?). It still had a rather serious bug in the buffer-cache handling code, but after patching that, it was pretty ok. 0.11 (December) had the first floppy driver, and was the point where I started doing linux developement under itself. Quite as well, as I trashed my minix386 partition by mistake when trying to autodial /dev/hd2.

By that time others were actually using linux, and running out of memory. Especially sad was the fact that gcc wouldn't work on a 2MB machine, and although c386 was ported, it didn't do everything gcc did, and couldn't recompile the kernel. So I had to implement disk-paging: 0.12 came out in January (?) and had paging by me as well as job control by tytso (and other patches: pmacdona had started on VC's etc). It was the first release that started to have "non-essential" features, and being partly written by others. It was also the first release that actually did many things better than minix, and by now people started to really get interested.

Then it was 0.95 in March, bugfixes in April, and soon 0.96. It's certainly been fun (and I trust will continue to be so) - reactions have been mostly very positive, and you do learn a lot doing this type of thing (on the other hand, your studies suffer in other respects :)

Linus

Appendix E - An insider's perspective of commercial DCE software.

From : Mark <cartermr@erols.com>
To : Tom Brennan <tomas_o_braonain@hotmail.com>
CC : <cartermr@erols.com>
Subject : Entera/DCE
Date : Wed, 20 Nov 2002 19:56:20 -0500

Hi Tommie,

First, I changed the subject so I don't accidentally delete a message in this thread. It looks like a spam (help wanted ;-)

Second, So sorry to hear about your job situation. I understand. The economy can't stay flat forever. Sounds like you're making the best of it by returning to school.

I tried to build some earlier free DCE distributions with little success. I have a Mac, and will not buy a WindozePC just for that. I don't see how a technology this complex can be free. Open source, perhaps. But not as a "self licking ice cream cone" like Perl or sendmail. It's just too big and bizarre. (No one system does what it did/could do to this day).

Entera 3.2 TCP (ever work with that?) load balanced very well. We tested it: I'm essentially a system and application admin and I don't trust our architects. We tested which server a client reached out to with individual calls (new bind each time). It balanced well, and paused only slightly when we forced server instances down. I've not seen where Entera 3.2 DCE had balance problems either. The client rips through it's partial bindings to connect to a server. We keep queue depth small and run lots of individual instances. So that if a client makes a call to a "busy" server, it goes elsewhere. There are other performance issues with the weird way Entera uses ObjectUUIDs when exporting to DCE namespace, but I can't say it's poor. I guess it depends on what you mean by poor. My take is that if a client favors an individual instance, that's poor. We don't have high overhead associated with locating a server for each RPC, because we bind during init and use the server until death (if we can).

Quotes are hard to find because, like airline seats, costs differ. I don't know what we pay for hardware/software. Not my job description. I know we pay for sw license, and did take out extended support from the vendors for dependent software (e.g., DCE itself). Right now, you can't get support for Entera 3.X anywhere I know of. Sorry I can't take the cost issues to ground. I do know it's getting expensive to maintain hardware/software (e.g., OS for those old machines) that still run Entera. We want to retire the services but the new ones (CORBA/WebSphere) aren't ready. That says something about the robustness of Entera and the DCE model.

Overall view: I liked working with Entera. It has idiosyncrasies, but so does CORBA and WebSphere. It was much better than native DCE API programming.

Once we built some test programs (to check how busy servers are) and monitored them all closely, we had a fairly stable environment. We seldom if ever lost the whole machine or entire enterprise. Contrast that with a rogue Java program that grabs all memory on the machine, or with a corrupted WebSphere admin server DB that would bring everything to a halt. We guarded our DCE cell, that's for sure. We used replication and backed up the pertinent DCE DB files, and used best practices to keep the cell stable. If DCE dies, Entera dies. But DCE servers were on dedicated machines and we had only one major outage when an undersized CDS server couldn't keep up with a large namespace. After we upgraded our server machines, so they can reload that DB into memory faster, we had no more issues. Knock on wood (the prod cell is still up).

I doubt we could have scaled up to an e-commerce environment under Entera. Not the way our systems were architected anyway. They say DCE scales, but the namespace resides in memory (cf LDAP) as does the registry. There were vendor plans to put a real DB on the backend, but by the time that may have happened, we froze our DCE version and moved on. I wasn't able to stay with the product then.

So, as an admin, I learned how to handle the Entera environment. I grew to appreciate it. It has shortcomings, and some of the programmers had issues,

but I can't know for sure if their problems were more in dealing with how we deployed the technology as opposed to inherent shortcomings.

Good luck, and if I can help further, let me know.

-Mark
cartermr@erols.com

> From: "Tom Brennan" <tomas_o_braonain@hotmail.com>
> Date: Wed, 20 Nov 2002 13:26:22 +0000
> To: cartermr@erols.com
> Subject: Re: Help wanted
>
> Hi Mark,
>
> Many thanks for your reply to my posting on google. Yes, you are correct
> about the feeling that DCE is dead - I have felt that when I embarked on
> this dissertation..., there is an open source version called freedce -
> http://sourceforge.net/projects/freedce
> Alas this version lacks security et al, but compilable under Linux. I have
> made a successful if not bare-bones model of a client/server. This software
> just creates an rpc portmapper daemon which sits in the background and
> execute the server, which listens for clients etc. You get the drift.
>
> The real reason is to evaluate freedce (from now on I'll refer to this)
> against the more commercial products. I have worked with Entera 3.2 under
> IBM RS/6000's AIX for nearly 6 years - 3 years on client/server development
> (but had to go back to college as of now, due to the slump in the I.T sector
> here in Ireland - we have had so many job losses here and had hard time
> trying to get a placement...anyway), but information is hard to get by on
> this one since borland/inprise have dropped support on entera 3.2, - I do
> remember it had poor balance-handling via round-robin scheduling - can you
> back me up on this?.
>
> Ideally, I would like to do some comparisons based on Entera w. FreeDCE,
> in terms of cost, licensing, cost of support per year. Borland/Inprise do
> not disclose the quotations of cost, which is what I'm trying to get my
> hands on. I am also looking into Entegrity www.entegrity.com which btw has a
> precompiled binary for various Linux distros.
>
> Am I correct in saying, that for Entera 3.2 for AIX, this would involve
> cost of support from Borland and from IBM per year, excluding licensing?
> Would you be able to help me out regarding costs of a) licensing, b)
> support, c) overall view of entera in terms of how you felt about working
> with it, easy etc..? I totally understand and do respect a company that is
> not willing to disclose the figures.
>
> Hope you could help me out.
>
> Thanking you,
> Tommie.
>
>
>
> _____
> MSN 8 helps eliminate e-mail viruses. Get 2 months FREE*.
> http://join.msn.com/?page=features/virus
>

Appendix F - A query posted to sourceforge regarding FreeDCE.

This has been edited for inclusion here in the appendix, the original posting can be found at
http://sourceforge.net/forum/forum.php?thread_id=760757&forum_id=29666

Project: Free DCE and DCOM: Forums: View Forum
Discussion Forums: Open Discussion Admin
By: tommieb_bsc (Tom Brennan)
Re: Where is http://dcerpc.net ????
Date: 2002-11-07 03:40
Hello again, where is the above website, I am getting desperate in looking for a most recent version of FreeDCE? Is there a mirror site available?

Thanking you,
T..

By: finieous (Phillip Iorio)
RE: Where is http://dcerpc.net ????
Date: 2002-11-11 13:15
It has been down for awhile now. What's up here is the most recent version. I've been working on some of the memory leaks, and some build fixes. Redhat 8.0 causes from breaks in the builds.

By: tommieb_bsc (Tom Brennan)
RE: Where is http://dcerpc.net ????
Date: 2002-11-11 15:00
Cool! I am working on a dissertation for my Bachelor of Science in Information Systems regarding Open Source vs Commercial Client/Server Software Development Technologies. I have tried to build Jim Doyle's port found at http://www.treepax.co.uk/FreeDCE which failed miserably due to missing bits of information in the instructions on that site so resorted to using this version here at sourceforge, which was very peachy and good, BTW I ran this under Suse 7.3 and had no problems....yet! The only thing that is missing is there is no means of setting up cells, security, authentication, nonetheless I have set up a client/server application successfully using the portmapper. What are the plans with this version of FreeDCE?
Like is security etc going to be put into this version?

Thanks!
Tom.

By: wez (Wez Furlong)
RE: Where is http://dcerpc.net ????
Date: 2002-11-11 15:27
freedce does not include those security services, and to implement them is one hell of a large project. The OSF have recently announced plans to release DCE/RPC 1.2 under an open source licence, and this is most likely your best bet for a full DCE/RPC solution.

By: tommieb_bsc (Tom Brennan)
RE: Where is http://dcerpc.net ????
Date: 2002-11-12 05:41
Many thanks for your response Wez and Phillip, Ok! I guess I am on my own here re: OSF/DCE, I've downloaded OSF/DCE 1.2.2, and the license states it is for academic/research only. Other than that, the makefiles only cater for AIX, DEC and HP-UX.....I am speculating would it compile if I change all references to it's own OSF/DCE dcethreads, and use this version.....food for thought.....Ho hum....Would you know of anyone who has used FreeDCE either in a company or personally for client/server development so I could find out what was their opinions on it were?

Thanks,
T.

This is another email I got from a guy called Bruce Foster, who answered my query at a newsgroup at google.com....and this was where I learnt of Entegrity...

From : Bruce Foster <bef@northwestern.edu>
To : "Tom Brennan" <tomas_o_braonain@hotmail.com>
Subject : Re: FreeDCE
Date : Mon, 4 Nov 2002 17:30:10 -0600
I think that the FreeDCE and Doyle stuff is pretty old and stale. The OpenDCE site just emerged, and at least the email list associated with it has generated some interest and traffic. If anything happens, it'll probably be from that. And some of the original DCE developers are active on that email list. In the long run, if you're going to do anything with the DCE code, you'll need to set up a security server and a CDS server. You can put them both on the same machine -- you can make your whole cell run on one machine for that matter. Of course, there are features such as replication that take advantage of multiple machines in a cell..

From what I recall, the development environment for DCE was quite unusual for its time, and it was not extended (modified) to fit current norms. Furthermore, it was not developed on Linux at all, so there are no hooks that help with Linux.

Doyle et al did a lot of work to get that environment to work somewhat on Linux. There was some chat about reconciling different thread environments too. The DCE threads are quite old Posix threads, and they are incompatible with modern thread libraries and calls. You'll find yourself working to prevent collisions. Commercial versions of the DCE services for Linux are under development at Entegrity Solutions (http://www.entegrity.com). You might check them out -- they may be willing to loan you their software for the duration of your project.

Bruce

> Hi Bruce,
>I have currently downloaded opendce 1.2.2,
>and obtained Jim Doyle's patches from a mirror
>site of boston uni (www.treepax.co.uk/FreeDCE),
>and have followed instructions, so far, it did
>not even get to the compile stage as the build
>scripts is broken.....I did however look at
>freedce from sourceforge.net/projects/freedce,
>and noticed something different about it....like
>I have tried to figure out how to set up a cell
>etc...as I am basing my dissertation on "open
>source vs commercial client/server s/w devel.
>tech" for Bachelor Science in Computers...this
>version from sourceforge simply compiles an rcpd
>- nothing more....no security set up etc...BTW I
>am running Suse Linux 7.2 w. 256M ram, 20G
>drive... here's hoping.....but so far it seems
>that the build scripts are a bit screwy....
>
>Thanks,
>T.

>>From: bef@northwestern.edu (Bruce Foster)
>>Reply-To: bef@northwestern.edu (Bruce Foster)
>>To: tomas_o_braonain@hotmail.com
>>Subject: Re: FreeDCE
>>Date: Thu, 31 Oct 2002 15:20:28 -0600 (CST)
>>
>>You should take a look at OpenDCE:
>>
>>http://www.opengroup.org/opendce
>>
>>Bruce
>>--
>>Northwestern University | Academic Technologies
>>| 1935 Sheridan Rd, 2East
>>Evanston IL 60208-2323 |
>>http://charlotte.at.northwestern.edu/bef/

Appendix G - A query in obtaining a quotation for Entegrity.

```
From :   "Tom Brennan" <93110376@lit.ie>
To :   tomas_o_braonain@hotmail.com
Subject :   Fwd: RE: Eval License Expiring
Date :   Wed, 11 Dec 2002 12:43:34 +0000
-----Original Message-----
From: Ray Cavanagh <ray.cavanagh@entegrity.com>
To: "'Tom Brennan'" <93110376@lit.ie>
Date: Fri, 6 Dec 2002 10:06:20 -0500
Subject: RE: Eval License Expiring

Tom - The current list price for Linux clients is $110 per license.
Support is 20% of list. So, for 50 users, the product cost is $5,500 and support is
$1,100. This is client only, the customer would nee do have the server side
to set up a cell. There may be charges for media and shipping as well.
There have been, literally, millions of lines of code written since the version
that FreeDCE represents. Let me know if you have any other questions.

Regards,
Ray Cavanagh
Director, DCE Worldwide Sales and Business Development
Entegrity Solutions
ray.cavanagh@entegrity.com
603-882-1306 x 2208

-----Original Message-----
From: Denise Cloutier
Sent: Monday, November 11, 2002 10:44 AM
To: 'Tom Brennan'; Denise Cloutier
Cc: Ray Cavanagh
Subject: RE: Eval License Expiring

        Hi Tom,

I forwarded your question to our Sales manger.  The support team will help
with any technical problems you may have with you Evaluation.

Ray can you answer this customer regarding pricing.

Regards,

   Denise P. Cloutier
   Support Administrator
   Phone 1-800-525-4343
   Fax 1-603-882-6092

-----Original Message-----
From: Tom Brennan [mailto:93110376@lit.ie]
Sent: Monday, November 11, 2002 5:33 AM
To: Denise Cloutier
Subject: RE: Eval License Expiring

Hi Denise,

    I am wondering would this be possible, I do not know how to ask…to
give me a pricing on Entegrity for say, avg 50 users in a company XYZ
running SuSE Linux 7.3, and price of technical support per annum? It
would be handy to get this info so that I can do broad conclusions like,
so far after carrying out research, there are some advantages with
commercial client/server devel. technologies over the likes of FreeDCE
which I was experimenting with....as a matter of interest what is the
cost for a single user re: same?

Thanking you,
Tom.

-----Original Message-----
From: Denise Cloutier <denise.cloutier@entegrity.com>
To: "'Tom Brennan'" <93110376@lit.ie>
Date: Tue, 5 Nov 2002 15:25:19 -0500
Subject: RE: Eval License Expiring

> That was suppose to be 'concerns'.. haha not concurs..
>
```

>
> Thanks,
>
>   Denise P. Cloutier
>   Support Administrator
>   Phone 1-800-525-4343
>   Fax 1-603-882-6092
>
>
> -----Original Message-----
> From: Denise Cloutier
> Sent: Tuesday, November 05, 2002 3:25 PM
> To: 'Tom Brennan'; Support Admins (suppadmin)
> Subject: RE: Eval License Expiring
>
> Surely,
>
> Feel free to contact us with any concurs. We would be glad to help!
>
>
> Cheers,
>
>   Denise P. Cloutier
>   Support Administrator
>   Phone 1-800-525-4343
>   Fax 1-603-882-6092
>
>
> -----Original Message-----
> From: Tom Brennan [mailto:93110376@lit.ie]
> Sent: Tuesday, November 05, 2002 2:18 PM
> To: Support Admins (suppadmin)
> Subject: Re: Eval License Expiring
>
>
> Hi Denise,
>
>   Many thanks for your email, but I have not tried it out yet as I am
> carrying out research as part of my Dissertation for Bachelor Science
> in Information Systems here in LIT, Limerick, Ireland, the dissertation is
> aptly called "Open Source vs Commercial Client/Server Development
> Technologies" and am currently researching into Jim Doyle's work on
> porting OSF/DCE across to Linux, I am not getting much success out of
> it so far.... I have yet to review any commercial DCE software....Entegrity
> is one of them..... I am wondering would it be ok should I have any
> questions about Entegrity, to contact you?
>
> Thanking you,
> Tom.
>
> -----Original Message-----
> From: "Support Admins (suppadmin)" <suppadmin@entegrity.com>
> To:
> Date: Tue, 5 Nov 2002 10:07:24 -0500
> Subject: Eval License Expiring
>
> >   Hi,
> >
> >   My name is Denise Cloutier - I am the support administrator for
> > Entegrity Solutions.
> >
> >   Recently you download an evaluation version of our product.
> >
> >   I just wanted to send a friendly reminder that your license is
> > about to expire.
> >
> >   We would be happy to prepare a quote for you, please let us know
> > how many PC-DCE Runtime, CDS Server, Security Server and ADK licenses you
> > need for your project.
> >
> >   Please send request to DCESales@Entegrity.com
> >
> > Regards,
> >
> >   Denise P. Cloutier
> >   Support Administrator

Bibliography

Bloomer, John, "Distributed Computing and the OSF/DCE", Dr. Dobb's Journal, February 1995, pg-18.
Borland, Borland's Entera 4.2 "Welcome Manual", available on internet - *http://info.borland.com/techpubs/entera/secure/entera/entera42/welcome/enterawel-42.pdf*
Cashin, John, "Client/Server Technology. The New Direction in Computer Networking.", ISBN-0-56607-008-2.
Chu, Chih-Ping, *"On the code development paradigm of RPC and Corba applications",*Computer Communications, 21, (1998), pg 267-278.
Cubranic, Davor, Open-Source Software Development, available on internet, last accessed, 13/11/2002, *http://sern.ucalgary.ca/~maurer/ICSE99WS/Submissions/Cubranic/Cubranic.html*
Davidson, John, "An Introduction to TCP/IP", Springer-Verlag, ISBN-0-387-93351-X.
Di-Bona, Chris, and others, "Open Sources; Voices from the Open Source Revolution", 1st Edition January 1999, ISBN - 1-56592-582-3.
Doyle, Jim, port of OSF/DCE for Linux, available on internet, h*ttp://www.treepax.co.uk/FreeDCE*
Drummond, Richard, Linux Format, The, Linux in Business, March 2001, available on internet, last accessed, 06/01/2003, *http://www.linuxformat.co.uk/archives/LXF12.business.pdf*
Entegrity, Linux DCE Client, available on internet *http://www.entegrity.com/products/dce/dce.shtml*
FreeDCE, available on internet, *http://sourceforge.net/projects/FreeDCE*
Free Software Foundation, available on internet, *http://www.fsf.org*
GPL, GNU, Software licence, available on internet, *http://www.fsf.org/licenses/gpl.txt*
Kozinski, Maciej, "Open Source - Ready for Business", available on internet, last accessed 12/11/2002, *http://newsforge.com/article.pl?sid=02/11/07/0246213*
Luce, Thom, "Computer Hardware, System Software, and Architecture", ISBN 0075577720.
Matthew, Neil/Stones, Richard, "Beginning Linux Programming", Wrox, Press, ISBN-1-874416-68-0.
Mockus, Audris, Fielding, Roy T., Herbsleb, James D., Open Source, a study, available on internet, last accessed 20/10/2002, *http://www.research.avayalabs.com/techreport/ALR-2002-003-paper.pdf*, *http://slashdot.org/articles/02/10/08/1553244.shtml?tid=99*
OSF/DCE, available on internet, *http://www.opengroup.org/dce*
OSF, The Open Group PDFs, *http://support.entegrity.com/private/doclib/indexDCE.shtml#osf122*
PSeries 610 Model 6E1, Entry level RS/6000, available on internet, last accessed 13/11/2002, *http://www-132.ibm.com/content/home/store_IBMPublicUSA/en_US/eServer/pSeries/entry/6106E1.html*
Ricciuti, Mike, "DCE: Yesterday's Technology for Tomorrow's Apps?", Datamation, July 1, 1994, pg 52-54.
Stallman, Richard, available on internet, *http://www.stallman.org/#serious* & *http://www.softpanaroma.org/People/Stallman*
Umar, Amjad, "Distributed Computing and Client-Server Systems", Prentice-Hall, ISBN-013-036252-2.

Index

(References)

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W