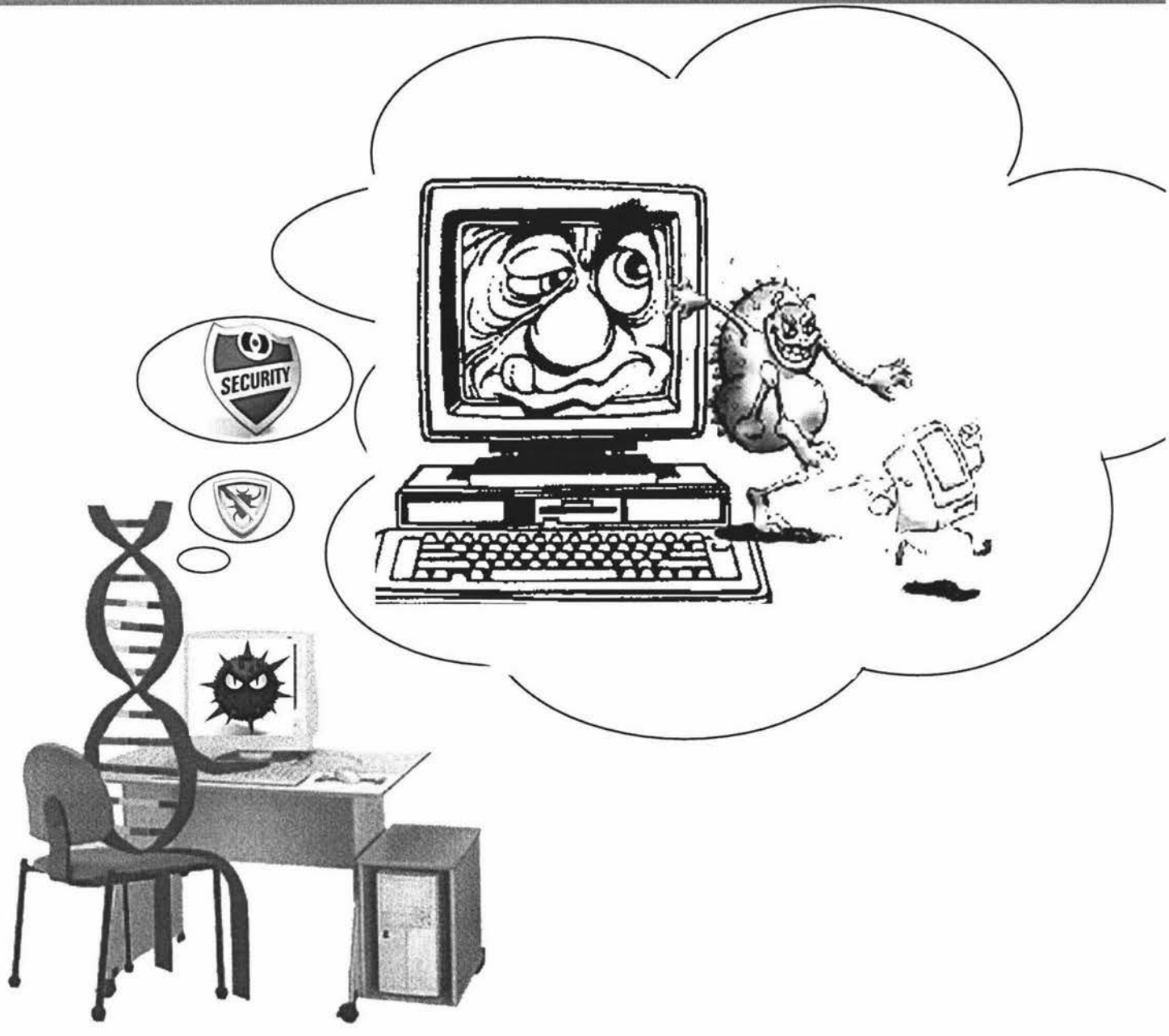
Massey University

Department of
Information Systems

157.899 Masters Thesis

Bio-mirrors and Networking Security



Prepared by **Douglas Mubayiwa**
Student 02430479

For the partial fulfilment of Masters of Information Sciences,
(Information Systems major), 2006

Abstract

Bioinformatics databanks have been the source of data to bioscience researchers over the years. They need this information especially in the analysis of raw data. When this data is needed, it has to be readily available. This thesis seeks to address the current problems of unavailable data at a critical time. Continued retrieval of data from far away sites is expensive in both time and network resources. Care must also be taken to secure this data otherwise by the time it reaches the researcher, it will be useless. In response to this problem being addressed, this thesis describes a way to move data securely so that the necessary data is stored nearest to whoever requires it.
A proposed initial prototype has been implemented with capacity to grow. The overall architecture of the system, the prototype and other related issues are also discussed in this thesis.

Acknowledgements

Research work of this magnitude cannot be created in isolation, especially when one is a family man, and I am grateful to many people for the help and support they rendered to me throughout the duration of this project. I would like to thank everyone who has contributed to the completion of this work, to come to mind pronto are:

• My supervisor, Sven Hartmann, for the much needed nudge when I needed wind to continue sailing, and for allowing me space when I needed it. Sven was my radar and I enjoyed his open door style. I learnt excellent management skills just by interacting with Sven. Thank you Sven.

• The staff of Massey University's Information Systems department, in particular, Markus and Madre , for providing advice and insight into how the systems at Massey University function, and for putting up with some of my more "interesting" ideas. "Oops! Did I say that?"

• Bryden for making sure I had no reason to complain about the department computers.

• My daughters for loving me even though I seemed to disappear when they needed me.

• Finally I would like to thank my wonderful wife, Rose, who has been my battle cry and primary source of inspiration over the years. Together through tight cooperation and a lot of sacrifices, we managed to come up academically, better than our starting point. We have shown that only through hard work and dedication can one's goals be achieved. Thank you Rose.

— Douglas Mubayiwa, May 2007

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background:

The objective of this research was to gain an in-depth understanding of the bioinformatics field and related topics. After this, I intended to develop a system that is able to recognize data movements between bio-mirror site and users. The amount of biological data is accumulating at an alarming rate, faster than improvements in computer hardware and current internetworking speeds. To make sense of complex biological systems, more data, often heterogeneous, should be included in increasingly complex computations. This data should be readily available otherwise the cost of retrieval will be too high. This thesis reviews the literature in the fields of Bioinformatics, Distributed Systems, Internetworking Security, and XML. A discussion on XML is featured prominently in this thesis because it is the vehicle chosen for biological data storage and distribution. It also has considerable impact on how a complete system should run. First this report will demystify bioinformatics issues before showing the proposed solution. It is hoped that the introduced system will inspire other developers to improve on it and make it an even better solution. The design and implementation details of the project are discussed to describe the developed prototype system. An in-depth analysis of the project's evaluation data is then given, concluding with a discussion of future work that has been identified.

1.2 Motivation:

In the biotechnology area, there is an ongoing need to share data that is stored and managed in publicly available web sites. The overall volume of the data stored in molecular biology databases has been growing tremendously, so much that transporting query results is hampered by existing relatively low Internet speeds. Currently, a major international project (called Bio-Mirror) is in place to provide for high-speed access to up-to-date molecular biology databases.

The idea is to establish local, regularly updated mirror sites that maintain local copies of molecular biology databases of interest for a country, region or community. Local bio-mirrors raise a number of security concerns for organisations running or using this service. I propose to investigate these concerns and to find pathways to address them. I will then describe the challenges I have encountered, particularly those relating to the long transaction times of bioinformatics resources and the difficulties of maintaining state, together with the solutions that I have developed to cope with these. High performance back-end computational infrastructure is essential for bioinformatics, but the cost of having the data to work with is equally critical.

1.3 Significance

Although the developments of relatively new Bio-Mirror (www.bio-mirror.net) archive services, such as to greatly reduce the bandwidth needs, while increasing their performance and usability to users, are well documented, this project should also have good impact, as a good solution would be adopted by most of these bio-mirror sites. This would reduce costs such as administrative by a great deal.

I believe also that adoption of this solution, albeit some changes here and there would result in a higher efficiency for repetitive high-throughput searches that result from the processing of large data sets. Much consideration has dwelt on such factors as the cost for storing and moving bioinformatics data. The subsequent cost of moving data from its new location to the end user is critical. We will discuss this in detail later on. In summary, we hope:

1. It will significantly reduce the time that a user must wait for requested data.
2. It considerably reduces the overall network traffic and servers processing power.
3. It will greatly reduces the load on the remote parent server, leaving it do handle other more critical tasks.

Web user interfaces for these sequence databases are well developed, we shall discus them, but wont direct much focus on these. For the purpose of the proposed system, it is sufficient to use the command line interface to issue commands and view outcome.

1.4 Related work

1.4.1 Hypothesis Research

The problem of searching huge biological databases on a large scale has been ongoing for several years. Biological databases are notorious for their massive volumes. Moving a file from its original location to your computer might require much in terms of storage space. Absolute care must be taken when selecting which data to move from a remote location and store locally. For the purpose of this research report, we shall not worry too much on the quality of data stored in such databases as GENBANK. We shall assume that this data is as accurate as it can be due to the measures kept in place at the point of entry. For our purpose, we shall also assume that all we are required to do is transport it as efficiently and as securely as possible. Primary focus will be on how this data gets to the final end user. In this day and age of disruptive man-in-the-middle, distributed computing data can reach the end user as useless piece of information that can seriously affect intended purpose like direction of study. Worse still, this could be data for medical purposes and might spell disaster to the end recipient (Kim, Choi, Hong, Kim, & Lee, 2003).

The bio-mirror project is doing a great job by trying to bring data as close to the end user as possible. More work however, needs to be done, in order to bring the 'right' data closer. The hypothesis of this research is that with careful and informed planning, we satisfy the customer more. We allow the customers to indirectly specify the data that they want and then fill a local mirror database with this invaluable data. We could start of with a near empty database that only contains information known to be important to the local community. Such information could be basic as competing organizations might choose to keep their need secretive for fear of giving away their cash cows. This will reduce the administration of the mirror site to low leaving the administrator to do other more demanding tasks. How this data is acquired is a subject of discussion on its own but for now, we must assume there is an aspect of secrecy where customers are kept from knowing each other.

To avoid filling up the database with information that is no longer required, we will assume we put checking measures to tell the administrator that particular data is now redundant. This data can then be cleaned up thereby freeing up invaluable space. Much related work to this is taking place in the research circles and a section dedicated to this will provide more detail.

1.4.2 Distributed Systems

The main ideas behind both bioinformatics and distributed systems computing are not new, and several successful bio-mirror systems and distributed systems have been developed on the basis of effectively distributing biological information in a timely manner (Strizh, 2006) (Bar-Or, Keren, Schuster, & Wolff, 2005). However, the current systems seem to focus only on the distribution of data. Much assumption is on the fact that this data reaches the end user as it was in the original state. Unfortunately, nowadays this is not the case anymore. In a dedicated section on Networking Security, we shall discuss the causes and cures of challenges.

1.5 Report outline

Following this introduction and quick review of related work, Chapter 2 of the thesis presents some essential bioinformatics and XML technology. Here is where we investigate the field of bioinformatics in a more detailed way. We also delve into XML schema extraction and illustrate some examples as to why it suits storage and in general the handling of bioinformatics data. Chapter 3 discusses bio-mirrors in detail and gives an example of one such site visited by this author during the study period. It starts off by exploring the major bioinformatics databanks and then look at the bio-mirror project itself. This bio-mirror project is, for the record, what we seek to enhance. Chapter 4 will focus on internetworking security. Issues such as the denial of service attack (DoS) and the address resolution attack (ARP) are covered to a great extend. We will also look at how security is provided at the physical layer and at the presentation layer of the OSI model, which itself would have been discussed, albeit, quickly. Chapter 5 will discus the implementation of the proposed system. Here, we will talk about the development language, the underlying physical and logical architectures as well as design decisions. Chapter 6 is basically about the test results. It is then followed by a discussion of future work in chapter 7 before the conclusion in the final chapter, chapter 8.

Chapter 2

Bioinformatics and XML

2.1 What is bioinformatics?

The National Centre for Biotechnology Information (NCBI 2001) defined bioinformatics as the field of science in which biology, computer science, and information technology merge into a single discipline as shown in figure 1. The ultimate goal of the field is to enable the discovery of the new biological insights as well as to create a global perspective from which unifying principles in biology can be discerned.

The Online Dictionary of Computing defines bioinformatics as: The field of science concerning the application of computer science and information technology to biology; using computers to handle biological information, especially computational molecular biology.

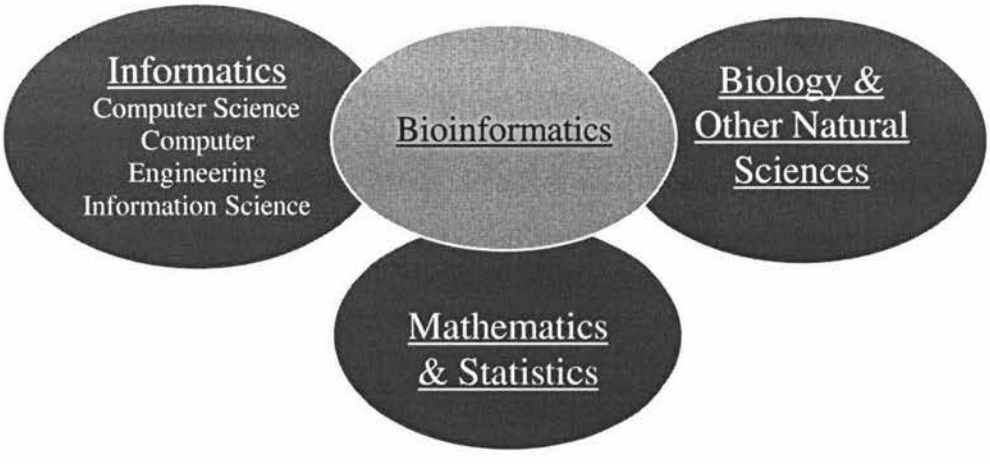


Figure 1: Bioinformatics in perspective.

Bioinformatics developers range from mathematicians, statisticians, computer scientists, to software engineers. There are people with all these skills combined and are known as Computational Biologists or put simply Bioinformaticians.

Bioinformatics data users are scientists from both industry and academia who use the tools created by the developers to extract the vast gene databases which have been created in the last two decades to extract useful information for a number of end-purposes (Arredondo et al., 2006).

Between these two (bioinformaticians and end-users) are databases, whose structure is often maintained by developers, but whose content is provided by users. The databases are predominantly government-funded and accessible to the public with a typical browser. Some users often copy their institution specific information from these public databases onto a local network computer to speed up access by their local users. This is where the concept of mirrors comes into play. We will discuss about this in detail later as it is the main focus of this report. Masyr was the first person to introduce the term 'bioinformatics' in 1989, in a journal called 'New Directions in Bioinformatics' (Masys, 1989).

2.1.1 What is Bioinformatics used for?

Bioinformatics data is used for a practically limitless number of tasks. Here, we will discuss about some of the most common uses:

(1) Examining the evolution of sequence across a phylogeny, to determine which portions have changed via genetic drift (generally these regions are "junk DNA" or of mediocre functional importance) and which have been changed or maintained by natural selection (these regions typically have high functional importance);

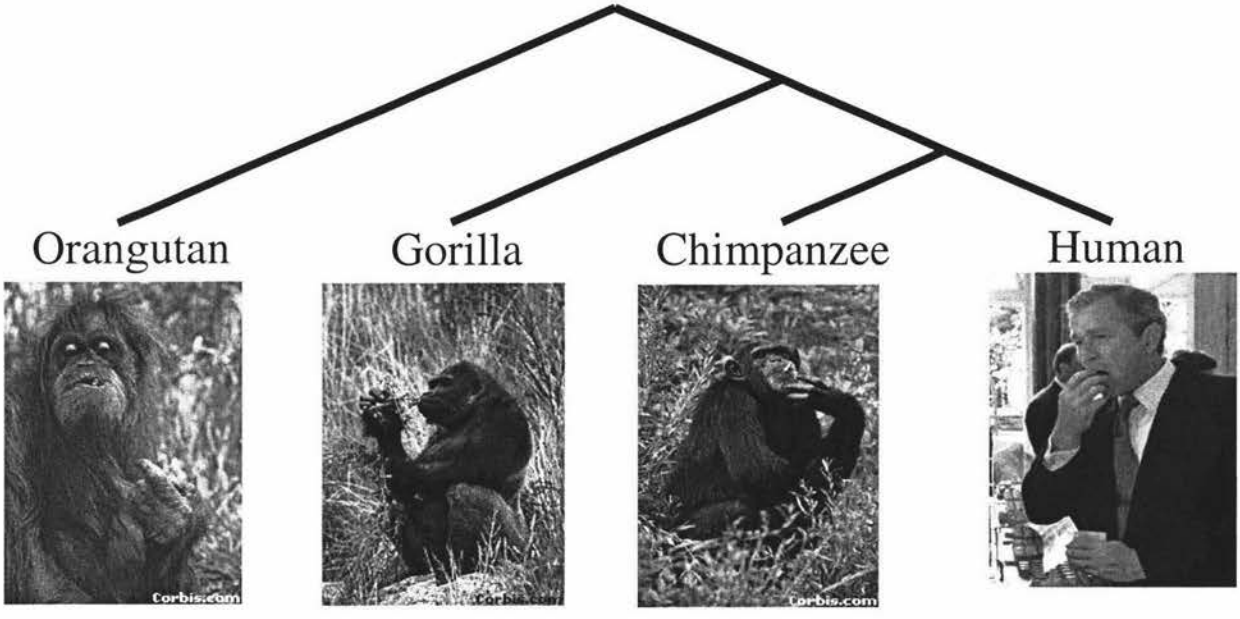


Figure 2: Tree of Life

(2) Comparing a new sequence to known sequences to determine if they are homologous and if so, similar structure and sometimes function is expected, and current knowledge can be applied to the new sequence. Several projects are underway, most notably the Tree of Life project being conducted by the University of Arizona. More information on this can be found on www.tolweb.org. Fig 2 is a summary of comparison of closely related species. Figure 4 will give a comparison of the Chimpanzee and Human gene structure. Notice any differences?

(3) Comparing whole genomes to examine large-scale patterns of evolution

(4) Reconstructing genes from EST sequences. **E**xpressed **S**equence **T**ags are short pieces of genes which are expressed, which have been cloned and sequenced - then deposited into the public gene databases. Given a 'zero database' - or no information to begin with, it is possible to reconstruct whole cDNA molecules by using regions of overlap to predict which sequences should fit together. Furthermore, given a known sequence from a different species, ESTs can be aligned against that template with their overlapping ends to allow the construction of a contig, or contiguous sequence.

(5) Grouping of proteins into families. There are is a huge amount of work being undertaken to classify the proteins encoded by genes into superfamilies and families. Protein sequences can be compared using an ever-increasing number of methods, their relatedness can be measured, and closely related proteins can then be assigned families. This is really an extension of (2), although grouping should involve the effective comparison of a sequence in GenBank with all the others. Since the number of sequences in GenBank is very large, this would take a long time, so clever

methods are used to model groups based upon smaller number of sequences. Pfam (Protein Families) is an extremely good example of the use of mathematics to group proteins into families and to make the information resulting from that both useful and accessible

2.2 Structure of bioinformatics data

Two important large-scale activities that use bioinformatics are Genomics and Proteomics. Genomics refers to the analysis of genomes. A genome can be thought of as the complete set of DNA sequences that codes for the hereditary material that is passed on from generation to generation. DNA exists as double helix of two conjugated strands and is treated as a one dimensional symbolic sequence made up of four letters from the alphabet (a, c, g, t) These DNA sequences include all of the genes (the functional and physical unit of heredity passed from parent to offspring) and the transcripts (the RNA copies that are the initial step in decoding the genetic information) included within the genome. Figure 3 is a three dimensional view of a DNA double helix to help visualise point. Thus, genomics refers to the sequencing and analysis of all of these genomic entities, including genes and transcripts, in an organism. Proteomics, on the other hand, refers to the analysis of the complete set of proteins or proteome (Chung & Wong, 1999).

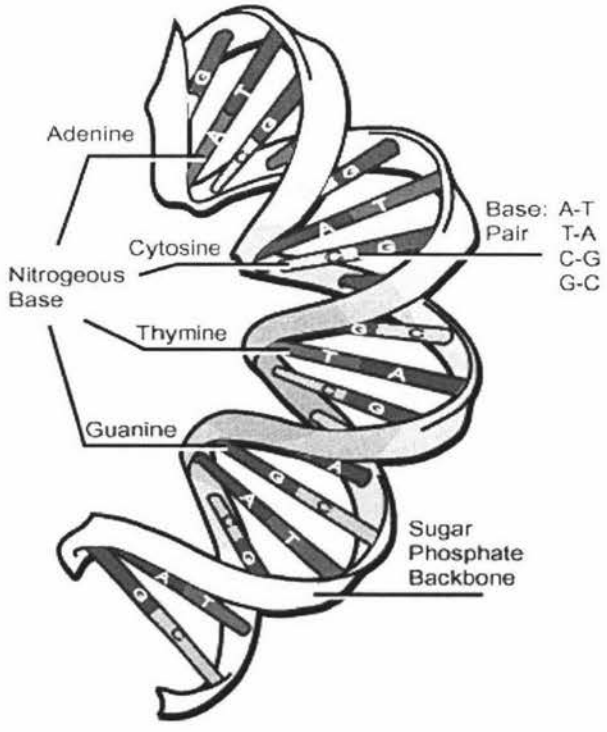


Figure 3: The DNA double helix.

1	ggcacattct cctgttgagc caggctatgc tgaccacaat gttgctgagc tgtgccctac
61	tgctggcaat gcccaccatg ctgggggccc agataggctt ggccccccctg gagggtatcg
121	gaaggcttga ccaagccttg ttcccagaac tccaaggtca gtgcgggcag gagtgggttg
181	ggtggggctt ggacatcctc tggccacaaa gtattctgct tgtatgagcc ctttcttccc
241	cttcccaatc ccaggcctgg gaggtgggtg ttttgtgcat gggtggttct gccctcacat
301	catctgtccc agatctaggc ctgcagcccc cactgaagag gacaactgca gaacgggcag
361	aagaggctct gctgcagcag gccgaggcca aggccttggc agaggtaaca gctcagggaa
421	agggctgagg ccacaagtct tgagtgggtg tgtcaagcat caacctctat ctgtgcttgg
481	agttgccact gtggtacaac gggattggcg gtgtcttggg agcgctggga cgtggtttca

Table 1: Pig (Sus scrofa agouti-related) protein gene

Table 1 shows the protein gene of a pig. To make more sense out of this, and for interest sake (as this report was compiled in New Zealand), below is a DNA fragment of the great spotted kiwi (Apteryx Haastii). This is the author's understanding of a question that was they attempted in a related paper. Attempt will be made to clarify this to a first time reader.

...GATCCTGACCATGAACCTAAGCTTCTTCGACCAATT...

The sequence of amino acids of a protein is encoded using triplets of bases, as shown in table 2, [A (Adenine), G (Guanine), C (Cytosine), T (Thymine)]. Decoding of a sequence of nucleotides requires that the sequence is spliced into triples (that is, blocks of 3 nucleotides) and then looked up from the table shown below.

	T	C	A	G	
T	Phe	Ser	Tyr	Cys	T
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	STOP	STOP	A
	Leu	Ser	STOP	Trp	G
C	Leu	Pro	His	Arg	T
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	T
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	T
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

Table 2: The Universal Genetic Code

There are three different reading frames possible after which repetition will begin, and two directions thus, the six candidate sequences of amino acids are shown in table 3... Direction 1

DNA string	GAT	CCT	GAC	CAT	GAA	CCT	AAG	CTT	CTT	CGA	CCA	ATT
AminoAcid	Asp	Leu	Asp	His	Glu	Pro	Lys	Leu	Leu	Arg	Pro	Ile

Sequence one = {Asp, Leu, Asp, His, Glu, Pro, Lys, Leu, Leu, Arg, Pro, Ile}

DNA string	G	ATC	CTG	ACC	ATG	AAC	CTA	AGC	TTC	TTC	GAC	CAA	T	T
AminoAcid		Ile	Leu	Thr	Met	Asn	Leu	Ser	Phe	Phe	Asp	Gln		

Sequence two = {Ile, Leu, Thr, Met, Asn, Leu, Ser, Phe, Phe, Asp, Gln}

DNA string	G	A	TCC	TGA	CCA	TGA	ACC	TAA	GCT	TCT	TCG	ACC	AAT	T
AminoAcid			Ser	stop	Pro	stop	Thr	stop	Ala	Ser	Ser	Thr	Asn	

Sequence three = { Ser, STOP, Pro, STOP, Thr, STOP, Ala, Ser, Ser, Thr, Asn }

... and reversed, direction 2,

DNA string	TTA	ACC	AGC	TTC	TTC	GAA	TCC	AAG	TAC	CAG	TCC	TAG
AminoAcid	Leu	Thr	Ser	Phe	Phe	Glu	Ser	Lys	Tyr	Gln	Ser	STOP

Sequence four = {Leu, Thr, Ser, Phe, Phe, Glu, Ser, Lys, Tyr, Gln, Ser, STOP}

DNA string	T	TAA	CCA	GCT	TCT	TCG	AAT	CCA	AGT	ACC	AGT	CCT	A	G
AminoAcid		STOP	Pro	Ala	Ser	Ser	Asn	Pro	Ser	Thr	Ser	Pro		

Sequence five = { STOP, Pro, Ala, Ser, Ser, Asn, Pro, Ser, Thr, Ser, Pro }

DNA string	T	T	AAC	CAG	CTT	CTT	CGA	ATC	CAA	GTA	CCA	GTC	CTA	G
AminoAcid			Asn	Gln	Leu	Leu	Arg	Ile	Gln	Val	Pro	Val	Leu	

Sequence six = {Asn, Gln, Leu, Leu, Arg, Ile, Gln, Val, Pro, Val, Leu}

Table 3: DNA sequence example

Direction One: From LEFT yields

Sequence one = {Asp, Leu, Asp, His, Glu, Pro, Lys, Leu, Leu, Arg, Pro, Ile}

Sequence two = {Ile, Leu, Thr, Met, Asn, Leu, Ser, Phe, Phe, Asp, Gln}

Sequence three = { Ser, STOP, Pro, STOP, Thr, STOP, Ala, Ser, Ser, Thr, Asn }

Direction Two: From RIGHT yields

Sequence four = {Leu, Thr, Ser, Phe, Phe, Glu, Ser, Lys, Tyr, Gln, Ser, STOP}

Sequence five = {STOP, Pro, Ala, Ser, Ser, Asn, Pro, Ser, Thr, Ser, Pro}

Sequence six = {Asn, Gln, Leu, Leu, Arg, Ile, Gln, Val, Pro, Val, Leu}

A DNA sequence identifies a given species, even those having the same nucleotide composition. This nucleotide structure is the most basic level of structure and hence defines the higher levels of the structure as well. After a DNA is sequenced, the next step is to find different functional regions in it as shown in table 4. The challenge is to gain more bio information from this. The case of the human genes and that of Chimpanzee as shown later in figure 2 is an interesting one. We share over ninety-five percent of our genes. Some part as the one explained in sequence comparison tools later, figure 3, is even more interesting, there is a full hundred percent in some functionalities.

A protein sequence is formed by twenty different kinds of Amino Acid. Amino acids are the basic structural units of proteins. An alpha-amino acid consists of an amino group, a carboxyl group, a hydrogen atom, and a distinctive R group bonded to a carbon atom, which is called the alpha-carbon because it is adjacent to the carboxyl (acidic) group (Varfolomeev, Uporov, & Fedorov, 2002). A complete list is shown in the table below but detail which can be found in various literatures is omitted. The reader is referred to (Varfolomeev et al., 2002) for a quick tutorial on the topic.

Amino Acid	3-letter code	1-letter code	Amino Acid	3-letter code	1-letter code
Alanine	Ala	A	Leucine	Leu	L
Arginine	Arg	R	Lysine	Lys	K
Asparagine	Asn	N	Methionine	Met	M
Aspartate	Asp	D	Phenylalanine	Phe	F
Cysteine	Cys	C	Proline	Pro	P
Glutamine	Gln	Q	Serine	Ser	S
Glutamate	Glu	E	Threonine	Thr	T
Glycine	Gly	G	Tryptophan	Trp	W
Histidine	His	H	Tyrosine	Tyr	Y
Isoleucine	Ile	I	Valine	Val	V

Table 4: Amino Acid codes

A typical database management system has powerful mechanisms for querying and retrieving data stored in tables, but is not ideal for searching through large strings, such as the one in the example above (the kiwi DNA structure). This implies that bioinformatics needs its own specific tools for data handling. Below, we will try and shed light into this.

2.3 Software tools for bioinformatics research

Due to vast amounts of available data, computer retrieval and analysis of biomedical data is becoming more important than ever. Many software tools have been developed for this purpose (Y. P. Chen, 2005). Software tools to facilitate research in Bioinformatics can generally be classified into four categories as 1. Data retrieval tools, 2. Sequence comparison tools, 3. Pattern discovery tools and 4. Visualisation tools. Below is a brief summary of what these are to aid the reader.

2.3.1 Data retrieval tools

The major tool for data retrieval is Entrez. This is an integrated data retrieval system developed by NCBI that provides integrated access to a wide range of data domains, including literature, nucleotide and protein sequences, complete genomes and related data. Each Entrez Gene record encapsulates a wide range of information for a given gene and organism. When possible, the information includes results of analyses that have been done on the sequence data. The amount

and type of information presented depend on what is available for a particular gene and organism and can include:

(1) graphic summary of the genomic context, intron/exon structure, and flanking genes,

(2) link to a graphic view of the mRNA sequence, which in turn shows biological features such as CDS and SNPs.

(3) links to gene ontology and phenotypic information,

(4) links to corresponding protein sequence data and conserved domains,

(5) links to related resources, such as mutation databases.

Entrez Gene is a successor to LocusLink (Liu & Grigoriev, 2005; Shah et al., 2005).

2.3.2 Sequence comparison tools

Basic Local Alignment Search Tool (BLAST) and FAST Alignment (FASTA) are the commonly used sequence comparison and alignment tools. BLAST is used for comparing gene and protein sequences against others in public databases, and now comes in several types including PSI-BLAST, PHI-BLAST, and BLAST 2 sequences. Specialized BLASTs are also available for human, microbial, malaria, and other genomes, as well as for vector contamination, immunoglobulins, and tentative human consensus sequences. FASTA can be used for a fast protein or nucleotide comparison. The program achieves a high speed by performing optimized searches for local alignments using a substitution matrix (Y. P. Chen, 2005). ClustalW is the tool of choice for multiple sequence alignments. http://www.ebi.ac.uk/clustalw/ contains more information on this.

The letters in figure 4 represent amino acids, the building blocks of DNA as discussed earlier. The sequences represent Cytochrome C, a gene which has to do with how the body uses oxygen. The upper set is the Chimpanzee's amino acids, while the lower set belongs to humans. Each "|" represent an amino acid match. In this case, Chimpanzee's and Human's have a 100% match of their Cytochrome C gene.

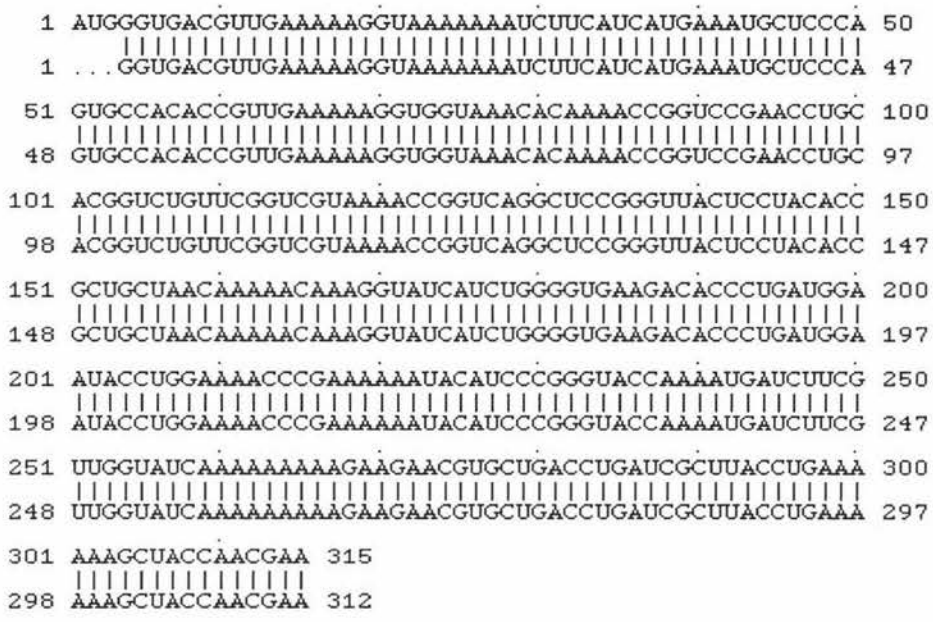


Figure 4: Chimpanzee and Human specific gene comparison

2.3.3 Pattern discovery tools

These are the tools that are used to search for patterns or features in the data. Cluster Analysis is the most common tool used for this purpose (Pinheiro et al., 2006; Xu & Wunsch, 2005). It is used to find groupings in a given dataset such that objects in the same group are similar to each other while those in different groups are not. Pattern discovery tools are also important in sequence analysis. They use advanced mathematical modelling and statistical interferences to find specific sub sequences, functional sites and structures (Park, Lee, & Park, 2005).

2.3.4 Visualisation tools

These allow an interactive graphical display of genomic data. Expression Profiler and GeneQuiz are examples and these incorporate a visualisation tool in them. Most visualisation tools are freely available on the internet; examples are TreeView, BioViews, and Protein Explorer. There is currently so much work to develop tools that can be used to interactively analyze the ever-larger data sets that result from continuing advances in bioinformatics (Davenport, Ellis, Ambrose, & Dicks, 2004; d'Inverno & Prophet, 2005; King, 2004). This subject on its own is rich in research.

2.4 Challenges with bioinformatics data.

Bioinformatics is a child prodigy, and in some ways is a victim of its own success. The size, complexity, and number of databases used in genomics and proteomics, as well as structure and other studies, have caused industry to lag behind in adapting to the need to handle these problems. Both computing technology and biology are developing new co-parenting techniques to create hopeful solutions, such as standardization, and a means of teaching their child (bioinformatics) to communicate better. Bioinformatics will undoubtedly, as any child ultimately does, find its own way, which means that many more problems lie ahead, yet there will likely be even more rewards to come (Lacroix, 2003).

Data management for molecular and cell biology involves the traditional areas of data generation and acquisition, data modeling, data integration, and data analysis. In industry, the main focus of the past several years has been the development of methods and technologies supporting high-throughput data generation, especially for DNA sequence and gene expression data (Richardson, Vanwye, Exum, Cowen, & Crawford, 2007; Thompson & Poch, 2006). New technology platforms for generating biological data present data management challenges arising from the need to capture, organize, interpret and archive vast amounts of experimental data. Platforms keep evolving with new versions benefiting from technological improvements, such as higher density arrays and better probe selection for micro arrays. This evolution raises the additional problem of collecting potentially incompatible data generated using different versions of the same platform, encountered both when these data need to be integrated and analyzed. Further challenges include qualifying the data generated using inherently imprecise tools and techniques and the high complexity of integrating data residing in diverse and poorly correlated repositories.

The data management challenges mentioned above, as well as other data management challenges (Richardson et al., 2007; Thompson & Poch, 2006), have been examined in the context of both traditional and scientific database applications. When considering these challenges, it is important to determine whether they require new or additional research, or can be addressed by adapting and/or applying existing data management tools and methods to the biological domain.

Dealing with data uncertainty or inconsistency for experimental data has required statistical, rather than data management, methods for example adapting statistical methods to gene expression data analysis at various levels of granularity has been the subject of intense research and development in recent years. The most difficult problems have been encountered in the area of data semantics properly qualifying data values for example, an expression estimated value and their relationships, especially in the context of continuously changing platforms and evolving biological knowledge. While such problems are encountered across all data management areas,

from data generation through data collection and integration to data analysis, the solutions require domain specific knowledge and extensive data definition and curation work, with data management providing only the framework example controlled vocabularies and ontologies to address these problems.

In an industry setting, solutions to data management challenges need to be considered in terms of complexity, cost, robustness, performance and other user and product specific requirements. Devising effective solutions for biological data management problems requires thorough understanding of the biological application, the data management field, and the overall context in which the problems are considered (Richardson et al., 2007; Thompson & Poch, 2006).

2.4.1 Problems in Bioinformatics

Integrated informatics is the dynamically construction of schemas to organise related cross-domain analysis results and background knowledge.

There are challenges in trying to achieve this as highlighted in figure 6. Some of the problems are:

- Too much un-integrated data from a variety of incompatible sources
- No standard naming convention is in place each with a custom browsing and querying mechanism (no common interface)
- Poor interaction with other data sources

The value of a particular dataset can be increased enormously by integrating the data analyses, at least at the interface level, with externally developed, structured information. Classically, there are several methods of accomplishing integration (Achard, Vaysseix, & Barillot, 2001):

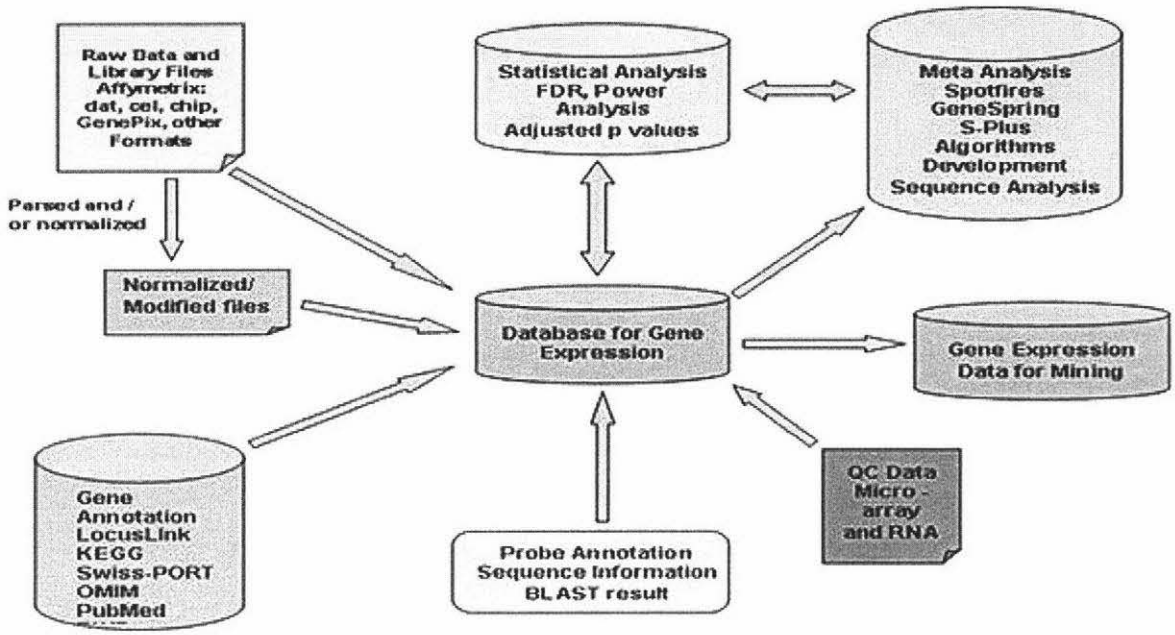


Figure 5: Data Integration

1. **Linked, indexed data systems**: Connect flat-file databases using web [HTTP] links and indexes;

2. **Loose integration:** Systematizing data into a multi-database system without a common schema but with a common query mechanism;

3. **Tight integration with views:** Organize them into a database association with a common schema and a central query mechanism;

4. **Loose integration with materialized data**: Organize them into a data warehouse without a common schema and periodically load all data into a central location.

5. **Tight integration with materialized data:** Construct a data warehouse with a common schema and periodically load all data into a central location.

These methods differ dramatically in their complexity of implementation and in their scalability and maintainability. They lie along a band, leading from complete physical and logical separation, to a logical union of physically incongruent datasets, and finally to an integrated whole.

In the long-term view, data integration is a key element to developing functional and useful bioinformatics systems. While it is in fact crucial that we archive and publish our data and results, the sheer number of important data types, as well as data-type-related databases is

inspiring. Raw data includes all types of nucleic acids, protein and hypothetical protein sequences, similarity results between and of these, functional data from a myriad of sources and varying protocols, genetic and physical maps, breeding data, associated environmental information and phenotypic information. Using computational tools to reduce complexity inherent to the reporting system, we begin to understand the dimensionality and complexity of the system which is truly of interest rather than focusing on details of the reporting method.

While the preceding has been, in a sense, a primer on the elements of bioinformatics systems, it has been important to present in the interest of developing a common basis for discussion of not just the current state, but the future needs and developments, of bioinformatics.

2.5 Extensible Markup Language (XML)

The Extensible Markup Language (XML) was created by W3C, the World Wide Web Consortium (Extensible Markup Language, 1998). It was specially designed to enable the use of large document management concepts for the World Wide Web that was embodied in SGML, the Standard Generalized Markup Language. In adopting SGML concepts however, the aim was also to remove features of SGML that were either not needed for Web applications or were very difficult to implement (de Knikker et al., 2004) (The XML FAQ, 2000). The result was a simplified dialect of SGML that is relatively easy to learn, use and implement, and at the same time retains much of the power of SGML (Achard et al., 2001; Barillot & Archard, 2000).

It is important to note that XML is not a markup language in itself, but rather it is a *meta-language* – a language for describing other languages. Therefore, XML allows users to specify the tag set and grammar of their own choice that follows the XML specification. This has enabled the bioinformatics community to exchange data objects such as sequence alignments, chemical structures, spectra etc., together with appropriate tools to display them, with more easily as compared to how they have done before using HTML. They now can use tags that their community can easily understand. Both Microsoft and Netscape support this new technology in their latest browsers (Bruhn & Burton, 2003) . XML is said to open the way to a new internet that would be easier to search and exploit. It gives the means for defining strongly structured document so computer programs can easily navigate through them and access relevant pieces of information. XML is often presented as a smart successor for HTML. It is a promising standard that is already undoubtedly impacting positively on the bioinformatics community.

2.5.1 XML Features

There are three significant features of XML that make it a very powerful meta-language (Bruhn & Burton, 2003), (Achard et al., 2001).

1. **Extensibility** - new tags and their attribute names can be defined at will. Because the author of an XML document can markup data using any number of custom tags, the document is able to effectively describe the data embodied within the tags. This is not the case with HTML, which uses a fixed tag set.
2. **Structure** – the structure of an XML document can be nested to any level of complexity since it is the author that defines the tag set and grammar of the document.
3. **Validation** – if a tag set and grammar definition is provided (usually via a Document Type Definition (DTD)), then applications processing the XML document can perform structural validation to make sure it conforms to the grammar specification. So though the nested structure of an XML document can be quite complex, the fact that it follows a very rigid guideline makes document processing relatively easy.

2.5.2 The XML Document

An XML document is a sequence of characters that contains *markup* (the tags that describe the text they encapsulate), and the *character data* (the actual text being "marked up"). **Error! Reference source not found.**x shows an example of a simple XML document.

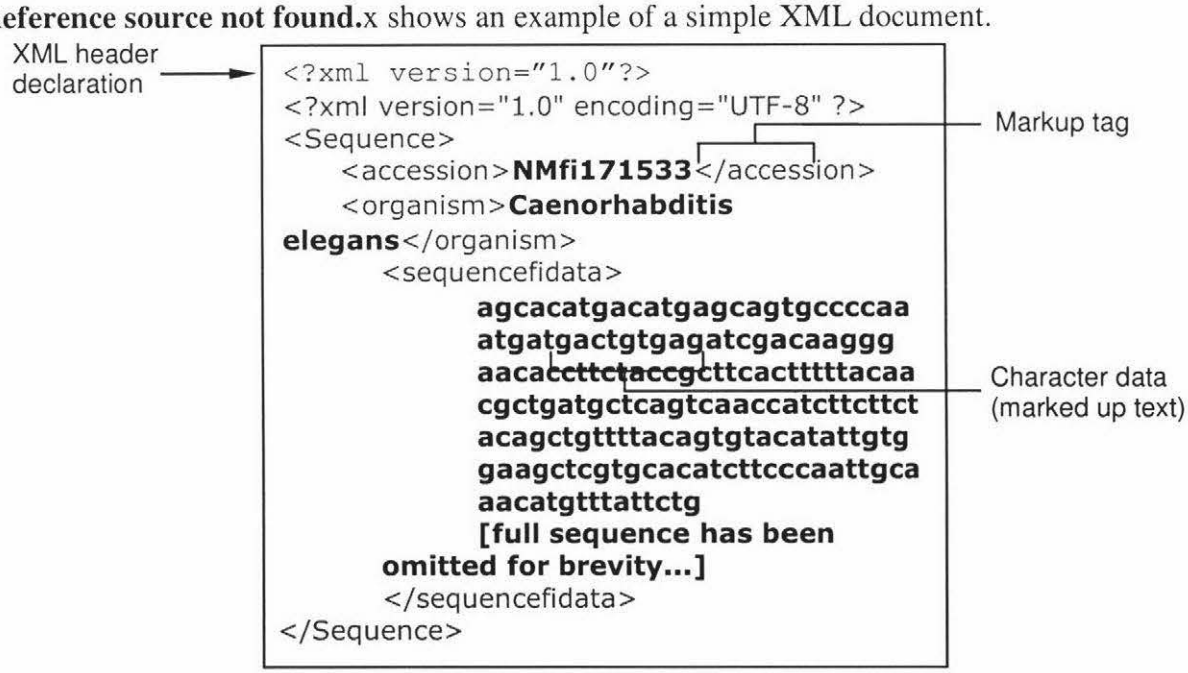


Figure 6: An XML document holding a nucleotide sequence record.

One of the main observations that should be made for the example given in **Error! Reference source not found.** is that *an XML document describes only the data*, and not how it should be viewed. This is unlike HTML, which forces a specific view and does not provide a good mechanism for data description (De Meo, Quattrone, Terracina, & Ursino, 2006). For example, HTML tags such as `P`, `DIV`, and `TABLE` describe how a browser is to display the encapsulated text, but are inadequate for specifying whether the data is describing a genetic species, a section of a patient's health record, or the price of a grocery item.

The fact that an XML document is encoded in plain text was a conscious decision made by the XML designers the designing of a system-independent and vendor-independent solution (Berman & Bhatia, 2005). Figure 6 shows an example of an XML document holding a nucleotide sequence record. Notice how easy it is to interpret. Although text files are usually larger than comparable binary formats, this can be easily compensated for using freely available utilities that can efficiently compress files, both in terms of size and time. At worst, the disadvantages associated with an uncompressed plain text file is deemed to be outweighed by the advantages of a universally understood and portable file format that does not require special software for encoding and decoding.

2.5.3 Benefits of XML

The following benefits of using XML in applications have been identified (Microsoft, 2000) (SoftwareAG, 2000b):

- **Simplicity** – XML is easy to read, write and process by both humans and computers.
- **Openness** – XML is an open and extensible format that leverages on other (open) standards such as SGML. XML is now a W3C Recommendation, which means it is a very stable technology. In addition, XML is highly supported by industry market leaders such as Microsoft, IBM, Sun, and Netscape, both in developer tools and user applications.
- **Extensibility** – data encoded in XML is not limited to a fixed tag set. This enables precise data description, greatly aiding data manipulators such as search engines to produce more meaningful searches.
- **Local computation and manipulation** – once data in XML format is sent to the client, all processing can be done on the local machine. The XML DOM allows data manipulation through scripting and other programming languages.
- **Separation of data from presentation** – this allows data to be written, read and sent in the best logical mode possible. Multiple views of the data are easily rendered, and the

look and feel of XML documents can be changed through XSL style sheets; this means that the actual content of the document need not be changed.

- **Granular updates** – the structure of XML documents allows for granular updates to take place since only modified elements need to be sent from the server to the client. This is currently a problem with HTML since even with the slightest modification a page needs to be rebuilt. Granular updates will help reduce server workload.
- **Scalability** – separation of data from presentation also allows authors to embed within the structured data procedural descriptions of how to produce different views. This offloads much of the user interaction from the server to the client computer, reducing the server's workload and thus enhancing server scalability.
- **Embedding of multiple data types** – XML documents can contain virtually any kind of data type such as image, sound, video, URLs, and also active components such as Java applets and ActiveX.
- **Data delivery** – since XML documents are encoded in plain text, data delivery can be performed on existing networks, sent using HTTP just like HTML.

Combined with the XML features discussed in section 3.5.1, the above list underscores the enormous potential of XML. Indeed, the extent of these benefits makes XML a core component in a wide range of applications: from dissemination of information in government agencies to the management of corporate logistics; providing telecommunication services; XML-based prescription drug databases to help pharmacists advise their customers; simplifying the exchange of complex patient records obtained from different data sources, and much more (Shabo, Rabinovici-Cohen, & Vortman, 2006).

2.5.4 DTD's and Validation

The XML specification has very strict rules which describe the syntax of an XML document – for instance, the characters allowable within the *markup* section, how tags must encapsulate text, the handling of white space etc. These rigid rules make the tasks of parsing and dividing the document into sub-components much easier. A *well-formed* XML document is one that follows the syntax rules set in the XML specification. However, since its author determines the structure of the document, a mechanism must be provided that allows grammar checking to take place. XML does this through the Document Type Definition, or DTD.

A DTD file is written in XML's Declaration Syntax, and contains the formal description of a document's grammar (The XML FAQ, 2000). It defines amongst other things: which tags can be used and where they can occur, the attributes within each tag, and how all the tags fit together.

Figure 7 gives a sample DTD section that describes two elements: `list` and `item`. The example declares that one or more `item` tags can occur within a `list` tag. Furthermore, an `item` tag may optionally have a `type` attribute.

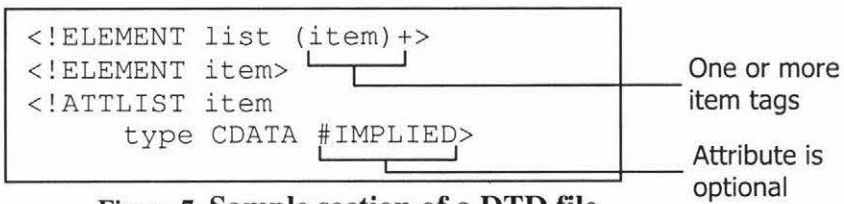


Figure 7: Sample section of a DTD file

Extending this example, the different levels of validation performed by an XML parser can be seen. Figure 8 shows an XML document that does not meet the syntax specified in the XML specification.

```
<?xml version="1.0"?>

<list><item>
      Item 1
</list></item>
```

Figure 8: XML syntax error - list and item tags incorrectly matched.

Figure 8 shows a well-formed XML document (that is, it follows the XML syntax), but does not follow the grammar specified in the linked DTD file. (The DTD file is the one given in Figure 7).

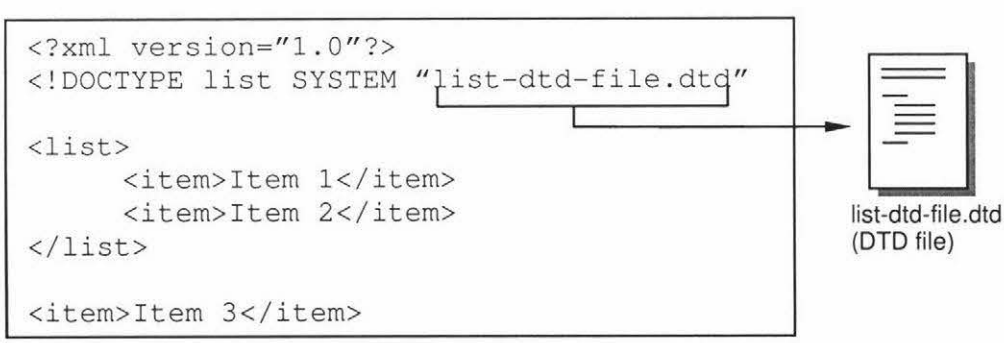


Figure 9: Well-formed XML document, but does not follow grammar specification in DTD file (an item tag occurs outside of list tag).

Figure 10 shows a well-formed XML document that also meets the grammar specification given in the DTD file.

```
<?xml version="1.0"?>
<!DOCTYPE list SYSTEM "list-dtd-file.dtd"

<list>
      <item>Item 1</item>
      <item type="x">Item 2</item>
      <item>Item 3</item>
</list>
```

Figure 10: Well-formed XML document that also follows DTD grammar specification. Will not produce any parse errors.

The XML Recommendation states that any parse error detected while processing an XML document will immediately cause a *fatal error* (Extensible Markup Language, 1998) – the XML document will not be processed any further, and the application will not attempt to second guess the author's intent. Note that the DTD does NOT define how the data should be viewed either. Also, the DTD is able to define *which sub-elements* can occur within an element, but not the *order* in which they occur; the same applies for attributes specified for an element. For this reason, an application processing an XML document should avoid being dependent on the order of given tags or attributes.

Chapter 3

Bio-mirrors

3.1 Bioinformatics Databanks

The first bioinformatics (biological) databases were constructed a few years after the first protein sequences began to become available. The first protein sequence reported was that of bovine insulin in 1956, consisting of fifty-one residues. Nearly a decade later, the first nucleic acid sequence was reported, that of yeast alanine tRNA with seventy-seven bases. Just a year later, Dayhoff gathered all the available sequence data to create the first bioinformatics database (Abbas & Holmes, 2004) .

The Protein DataBank followed in 1972 with a collection of ten X-ray crystallographic protein structures, and the SWISSPROT protein sequence database began in 1987. A huge variety of divergent data resources of different types and sizes are now available either in the public domain or more recently from commercial third parties. All of the original databases were organised in a very simple way with data entries being stored in flat files, either one per entry, or as a single large text file (Y. P. Chen, 2005). Re-write-Later on lookup indexes were added to allow convenient keyword searching of header information. Bioinformatics data across diverse databases is often highly redundant, mainly because benchmarking and quality control mechanisms are rudimentary (Y. P. Chen, 2005).

We will give a brief outline of the International Nucleotide Sequence Database Collaboration (INSD). It is a cooperation of the three well known databases EMBL Bank, GenBank and DDBJ. These three organizations have separate sites for data submissions, and they exchange data on a daily basis. This implies that despite a variation in style and annotation format, the sequence information in the three databases is the same at any given time. They have a board namely the International Advisory Committee whose main responsibility is to endorse and reaffirm the existing data sharing policy of the three separate databases(Okubo, Sugawara, Gojobori, & Tateno, 2006). The INSD is in widespread use by the biological community. Information used

here is from their individual websites. There exists a policy which must be adhered to by individuals submitting data to the databases. Figure 11 below was sourced from their webpage.

Figure 11: INSDC (source insdc.org)

3.1.1 GenBank: Location USA.

Genetic Sequence Databank, GenBank (http://www.ncbi.nlm.nih.gov/Genbank/) is the National Institute of Health (NHI) genetic sequence database and is an annotated collection of all publicly available DNA sequences. It is one of the fastest growing repositories of known genetic sequences. Figure 12 (courtesy of Genbank) shows the exponential nature of the growth of the bank. GenBank is part of the International Nucleotide Sequence Database Collaboration, which is comprised of the DNA DataBank of Japan (DDBJ), the European Molecular Biology Laboratory (EMBL), and itself, GenBank at the National Center for Biotechnology Information in the United States of America. (Bilofsky, 1986; Cochrane et al., 2006). Each GenBank entry includes a concise description of the sequence, the scientific name and taxonomy of the source organism, and a table of features that identifies coding regions and other sites of biological significance, such as transcription units, sites of mutations or modifications, and repeats.

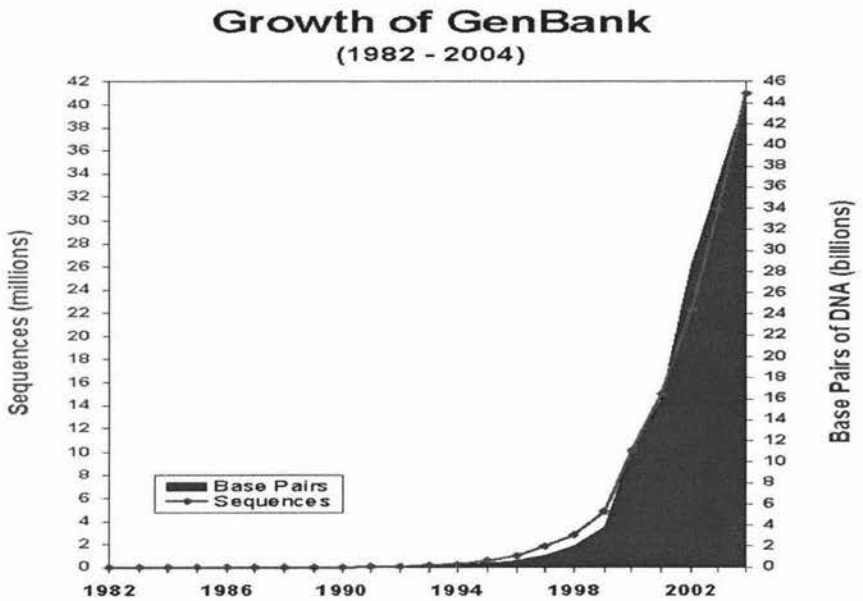


Figure 12: Growth of GenBank

3.1.2 DNA DataBank of Japan (DDBJ)

DDBJ (http://www.ddjb.nig.ac.jp/Welcome-e.html) began DNA data bank activities in earnest in 1986 at the National Institute of Genetics (NIG). DDBJ has been functioning as the international nucleotide sequence database in collaboration with EBI/EMBL and NCBI/GenBank as stated in the description above. DNA sequence records the organismic evolution more directly than other biological materials and, thus, is invaluable not only for research in life sciences, but also human welfare in general. The databases are, so to speak, a common treasure of human beings. With this in mind, these databases are accessible online freely and all the time.

3.1.3 EMBL: Location Europe

The EMBL Nucleotide Sequence Database (http://www.ebi.ac.uk/embl/index.html) is a comprehensive database of DNA and RNA sequences collected from the scientific literature and patent applications and directly submitted from researchers and sequencing groups. Data collection is done in collaboration with GenBank (USA) and the DNA Database of Japan (DDBJ). The database currently doubles in size every 18 months and as of June 1994, contained nearly 2 million bases from 182,615 sequence entries. The EMBL Nucleotide Sequence Database which is also known as EMBL-Bank constitutes Europe's primary nucleotide sequence resource.

After the discussion on XML, a DTD will be shown in the appendix section to help describe the structure of entries in the EMBL database (Cochrane et al., 2006).

3.2 Bio-mirror project

The Bio-Mirror project was formed in 1998 as a collaboration of Asian-Pacific bioinformatics Centers (APBionet), and IUBio Archive at Indiana University (Gilbert et al., 2004). This project's primary goal is the efficient distribution of large scale genomic data sets to various mirror sites around the world. As multi-gigabyte public bioinformatics databanks grow and change daily, access to them is hampered by limits on Internet bandwidth. The Bio-Mirror project addresses this problem with rapid redistribution from several sites. Such mirrors reduce the burden on source providers, and mitigate Internet outages and slow distant connections (Gilbert et al., 2004). It distributes databanks using high internet connections between continents. Many APBionet members are part of new and growing bioinformatics centers, where Bio-Mirror sites provide a short path to current data from the United States of America and various other European sources. Project sites serve data to a range of educational, government and industry bioinformatics groups. High speed access between the sites is provided by the network infrastructure developed by very High Speed Backbone Service (vBNS), TransPAC (Trans-Pacific network), and Asia-Pacific Advanced Network (APAN), and others. These sites are well connected to national research and education networks within each country (*Biomirror Public Access*).

The Bio-Mirror project is provided as a public service by member centers with support of several organizations. Other bioinformatics and public data services are invited to participate, and to utilize this high-speed access to data sets. No association with APAN is expected of bio-mirror project members. The table 5 below represents a summary of the current member sites.

Country	Host	Web site	Bulk data access
Australia	Australian National University	--	rsync: and ftp://bio-mirror.-au.apan.net/biomirror/
China	Institute of Microbiology, Chinese Academy of Sciences	http://bio-mirror.cn.apan.net/	ftp://bio-mirror.cn.apan.net/-pub/biomirror/
Japan	Computer Center for AFFRC	http://bio-mirror.jp.apan.net/	ftp://bio-mirror.jp.apan.net/-pub/biomirror/
Korea	Korea Advanced Institute of Science and Technology	http://bio-mirror.kr.apan.net/	ftp://bio-mirror.kr.apan.net/-pub/biomirror/
Malaysia	Universiti Putra Malaysia	http://ingene2.upm.edu.my/	ftp://ingene2.upm.edu.my/
Singapore	National University of Singapore	http://bio-mirror.sg.apan.net/	ftp://bio-mirror.sg.apan.net/-biomirrors/
Taiwan	National Yang-Ming University	http://bio-mirror.ym.edu.tw/	ftp://bio-mirror.ym.edu.tw/-biomirror/
Thailand	Kasetsart University	http://bio-mirror.ku.ac.th/	http://bio-mirror.ku.ac.th/-biomirror/
USA	IUBio Archive, Indiana University	http://www.bio-mirror.net/	rsync: and ftp://bio-mirror.net/-biomirror/

Table 5: Bio-Mirror sites.

3.3 Case Study (the Auckland University Bio-Mirror site)

3.3.1 Overview of setup:

Data that was made available to this author as of the 29th of September 2005 indicate that the University host over three hundred Gigabytes (351 GB) of biological data on their bioinformatics server. This data is mirrored every night from the Australian Bio-mirror, which in turn mirrors the US Bio-mirror.

The University uses a program or rather a service called 'rsync' for the mirroring. As has been described before, rsync compares files and keeps track of changes, and as such only downloads files that have been updated.

At the moment Auckland University Bio-mirror Centre is in the process of moving the NZ Bio-mirror to a dedicated server, as it is currently hosted on a server that is used for other purposes. The new hardware will be an IBM H20 Blade Server with two 3.0 GHz central processing units (2*3.0 GHz CPUs), which is hoped, will be more than capable of handling the load especially given its exponential increase.

The server runs on Redhat Enterprise Linux 3.0 operating system. For storage, use is made of Auckland University's Server Area Network (SAN) system, and currently an allocation of 500Gb is available to the Bio-mirror project.

The nightly updates account for anywhere between 4 and 10 GB of updated data. This however does not imply that the Bio-mirror grows by 4 to 10 GB per night. Most of the time files get replaced completely with slightly larger updated files, and rsync has the capacity to remove obsolete files as well, so the total size of the Bio-mirror data can fluctuate. A week prior to the release of this data, the total size was 360 GB and it should be noted here that it fluctuates down sometimes as well.

For various reasons and especially to do with bandwidth and financial cost the university had to set a strict time-frame within which the nightly updates are allowed to occur. Currently, the time-frame is from 8pm to 6am.

Not all the 'packages' (data-sets) hosted on the Australian Bio-mirror are mirrored, only the ones that the University feels are essential. It should be again noted that the Australian Bio-mirror hosts a number of packages that are not hosted by other Bio-mirrors, and those packages are mirrored on the NZ server. It is the University's intention to add more packages, but due to various other challenges as will be outlined later in this case, that will have to wait until, at least, after the new dedicated server has been installed.

Updates are grouped by packages, that is, each package has a separate update process. There is a substantial difference in both the size, importance and the update frequency of the various hosted packages, so the update processes have been scheduled to account for this within the nightly set timeframe.

On the 28th of October 2004, the total size of the NZ Bio-mirror was 257 GB, and as of the 29th of September 2004, it had increased by about 100 GB. This works out to an approximate increase in size of 334 days/ 100Gb = 0.3 GB per night. This year, traffic going into and out of the server has been 1.5TB and next year, it is projected to be 2.6TB. This projected traffic is estimated to be 6% of the total University of Auckland traffic, and is estimated to cost approximately $54,000. This estimated amount, of course, does not include overheads associated with administration, storage, air conditioning, and others. Uplink and downlink data is charged alike.

3.3.2 Some of the problems that have been encountered

3.3.2.1 Problems with overseas bio-mirrors:

The nightly updates are automated. Nobody really monitors them. If some files get corrupted on the US Bio-mirror, then that will propagate to the Australian Bio-mirror, and in turn end up on the NZ Bio-mirror. It can take quite some time before mistakes are detected.

The central/master US Bio-mirror is in essence a 'collection point' for data from 10-20 'source' servers spread all over the US. It mirrors a large number of packages into one place, which is then in turn mirrored in its entirety by other bio-mirrors. If changes are made on these 'source' servers, people who get their data directly from a bio-mirror might occasionally encounter surprises. For instance, one of the university's regular users found that a large group of files they were relying on for their research had suddenly disappeared, because somewhere down the line the data was no longer provided or no longer part of the mirroring process. This implies that it is left up to the users to monitor the package providers' websites for news and updates.

3.3.2.2 Problems with rsync and bandwidth:

If the Australian Bio-mirror is unavailable the university cannot 'switch' their mirroring to the US Bio-mirror, basically for two reasons namely:

1. If they were to switch mirrors, the rsync program would 'detect' that all files had been updated, and it would try to download all 400+ Gigabytes of data on the US Bio-mirror, as opposed to only the files that have changed.

2. Bandwidth to the US Bio-mirror is shockingly bad. It would not be able to complete the nightly updates in the timeframe set aside for them. Most Bio-mirror servers around the world appear to mirror the US Bio-mirror directly, so it is only fair to go so far as to say the US Bio-mirror is probably slightly overloaded. This is the primary reason why the University does not mirror directly from the US mirror. It is far too slow.

3.3.2.3 Problems with storage:

The University anticipated that the total size would double every year, but so far they have been pretty lucky. When they first set up the Bio-mirror they had only allocated 300Gigabytes of space which they quickly ran out of. At that point they dropped a number of packages that they felt needed no mirroring. Sooner or later however, there will be need to once again have to look at increasing the amount of storage available on the bio-mirror server.

The university purposely chose not to make backups of their bio-mirror. This is due to the fact that data is mirrored every night, so they did not think this was necessary. This does mean however that if there is corrupted data on their server, and the Australian site is unavailable, then they can not restore to a known 'good' backup. They would have to wait until the Australian Bio-mirror is back online and fixed. This scenario occurred between the months of July and August 2005.

3.3.2.4 Problems with access and a touch of security:

The NZ Bio-mirror project has restricted access to the bio-mirror site to New Zealand access only, and only allows people outside Auckland University access after 8pm. Unfortunately there appear to be quite a few networks/servers in New Zealand that have an incorrectly setup DNS. This means that their ip-addresses do not resolve to New Zealand addresses and thus the University's server blocks access. They have had to make access exceptions for troublesome servers and networks and they might have to consider removing the NZ only access restriction.

Within any university setup as in the case of Auckland University, access can be very open as much as a university can be. The majority of Windows Operating systems machines on the network are highly susceptible to all sorts of attacks in spite of increasingly intense efforts to protect them. Infected machines create network traffic, and we see performance dips because of that traffic. In order to protect our lab from threats, a number of security measures are in place, in addition to those described above.

To enhance security, access is controlled by associating specific IP addresses to specific resources on the workstations and servers. Control of that access is primarily through the use of ip tables. The server firewall allows incoming SSH traffic from anywhere. It then performs IP address filtering to allow only certain IP addresses access to more open resources, such as NFS,

LDAP, CUPS and the FlexLM license server. The Web server uses a slightly different setup to allow only incoming SSH and HTTP traffic. Each of the workstations has even more restrictive settings to allow only incoming SSH and VNC traffic. Outgoing traffic on all the machines is considered to be secure.

Each of the services allowed uses standard ports listed in /etc/services with the exception of NFS and FlexLM. FlexLM port usage can be controlled by modifying the license file by adding a specific port to use. NFS is a little more troublesome. In order to get NFS set up to use static ports, some background information is needed on how it works. For a detailed description, one can refer to the LinWiz documentation. The LinWiz site also includes a Web app that allows you to create an iptables configuration you can use as a starting point for your firewall.

3.3.1 Conclusion of case study:

This case study was conducted by a series of emails and a subsequent visit to the university's bio mirror site by this author. It is my belief that some of the problems that Auckland University (AU) is facing can be greatly reduced if Massey University (MU) mirrors their site. For instance, the fact that AU is hard on storage space which has forced them not to make nightly backups of their bio-mirror site implies there would be several instances where no service is available in New Zealand as a whole. This has been outlined under problems with storage. AU takes all its data from Australia which balloons the cost. The two universities could then share the cost of the mirroring. This way, MU students would benefit greatly from hands on experiments with bioinformatics data. Like with other university departments, this critical field can be easily offered at undergraduate level so as to arm students with more life options.

In future, this author is planning to build a mirror site predominantly of the AU bio mirror site. This would imply a last known good local backup that could be easily called up in case of emergency. Apart from this, it is planned that the site will later mirror and store specific data to New Zealand that is spread all over the world. The kiwi species that was discussed before, for instance, come to mind straight away. A quick random check for the strand resulted in visiting a site in Sweden. It would be nice to readily find accurate and up-to-date data of this endangered national symbol locally.

Chapter 4

Internetworking Security

Internetworking security is an increasingly important factor in the field of bioinformatics because of the central role the online databases we touched on before, various applications and groupware such as e-mail play an important role in the day-to-day operation of a bioinformatics facility. Opening an intranet to the outside world through username and password-protected restricted access may be the basis for collaboration as well as a weak point in the security of the organization (Hoschek, Jaen-Martinez, Samar, Stockinger, & Stockinger, 2001; Sun, Shi, Liu, & Sclabassi, 2005). In addition, because many biometric laboratories are involved, with applied genomics, there is a group of politically active opponents to this research field. The computer-savvy members of these activist groups represent a potential threat to network security. Some hackers do this just for fun.

Every network presents a variety of security holes through which potential hackers and disgruntled or simply curious employees can implement random threats, such as viruses. Many of these threats are network and operating system specific. For example, Microsoft typically announces a service pack within a few weeks after the introduction of a server-based operating system to patch security holes discovered by users. Cisco Systems are also constantly upgrading their Internetworking Operating System better known as IOS to stay ahead of the game.

The most secure method namely physical isolation from outside networks is not usually a viable option. Physical isolation is very expensive to implement. Even a considerably closed network without dial-in or any other wired access to other networks can be breached by persons with enough motivation and time. Every cable, peripheral, and display device emits a radio frequency signal that can be captured, amplified, and read. For this reason, computer facilities used by military contractors are frequently located in shielded, windowless rooms that minimize the chances of the radiation emitted from a computer reaching someone who is monitoring the building. Special types of cabling are also used to counter this. The diagram below shows the difference between Unshielded Twisted Pair (UTP) and Shielded Twisted Pair (STP). The main

difference is that with STP, reject better the effects of internal or external generated EMI/RF (Electromagnetic Interference/Radio Frequency).

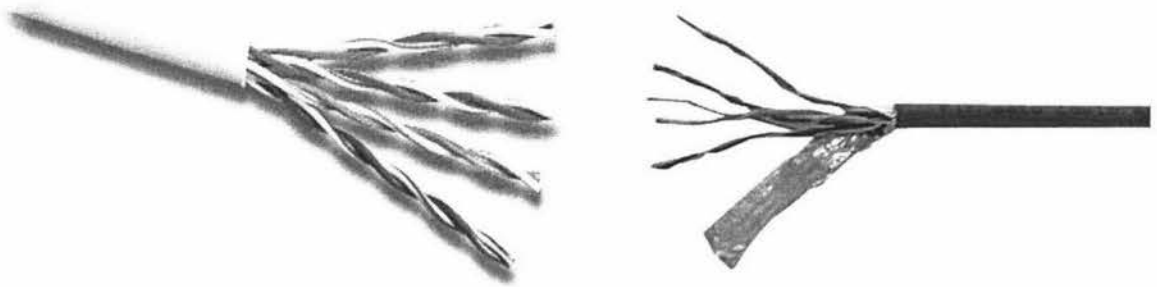


Figure 13: Unshielded Twisted Pair (UTP) cable left and Shielded Twisted Pair (UTP) cable right

Although it may be practically impossible to maintain security from professional industrial spies, a variety of steps can be taken to minimize the threat posed by modestly computer-savvy activists and the most common non-directed security threats. These steps include using antiviral utilities, controlling access through the use of advanced user-authentication technologies, firewalls, and, most importantly, low-level encryption technologies.

4.1 Denial of service attack (DoS)

Something that is harder to control would be a flood of pointless requests. We point to the DoS as the main enemy of the proposed system. During the implementation of the IMS, we have relied on what people term the ideal world where people leave in harmony with each other. We however accept that today's world is far from it, as not everybody thinks constructively. In the IMS, requests are coming from single clients and the nature is predetermined. Pointless requests would hit such a proposed system badly than some systems because this particular one has no built in caching system, if many identical requests are made, only a couple of the first ones will trigger a costly retrieval and mirroring sequence with all subsequent identical requests directed to local bio-mirror servers thus reducing the cost. If a large set of requests are made using different and valid identifiers, there could be a problem with overloading the parent site. This would be quite hard to solve, but monitoring of the frequency of requests (possibly taking note of IP

addresses) might allow certain IP addresses and domains to be barred, or allow the queue to be paused while the counterfeit jobs are removed (Chang, 2002). Note that this situation may not be malicious – it may be the result of automated job submission by a user who wishes to process a large number of requests. The solution to this case may well be to make it clear on the site that this is not considered acceptable use. This also needs more attention in the future and is a subject of further research. At this point we will take the opportunity of explaining a bit about denial of service.

So, tell me what it is?

A Denial of Service, or DoS as it is often abbreviated, is a malicious attack on a network. This type of attack is essentially designed to bring a network to its knees by flooding it with useless traffic. Many DoS attacks work by exploiting limitations in the TCP/IP protocols (Y. Chen & Hwang, 2006).

Hackers use DoS attacks to prevent legitimate uses of computer network resources. DoS attacks are characterized as attempts to flood a network, attempts to disrupt connections between two computers, attempts to prevent an individual from accessing a service or attempts to disrupt service to a specific system or person. Those on the receiving end of a DoS attack my lose valuable resources, such as their e-mail services, Internet access or their Web server (Jeong, Kim, & Kim, 2006). Some DoS attacks may eat up all your bandwidth or even use up all of a system resource, such as server memory, for example.

A DoS attack may very well appear to be genuine traffic on the network, just like normal queries from clients, but differs in that the volume and frequency of the traffic will increase to unmanageable levels. An attack on a bioinformatics server, for example, would not be normal queries, but rather a large onslaught of hits in close proximity so the server cannot keep up with the sheer volume of requested information. The targeted server would most likely be brought to a halt from a DoS attack because it runs out of swap space, process space or network connections (Kumar, 2005).

DoS attacks do not usually result in information theft or any security loss for a company, they can cost an organization both time and money while their network services are down (Jeong et al., 2006). Early DoS attacks consisted of simple tools generating packets from a single source which was then aimed at a single destination. The evolution of the DoS attack however now sees single source attacks against multiple targets, multiple source attacks against single targets, and multiple source attacks against multiple targets(Ayres, Sun, Chao, & Lau, 2006; Chan, Chang, Lu, & Ngiam, 2006).

4.1.1Common DoS Attacks

4.1.1.1 Buffer Overflow

The condition wherein the data transferred to a buffer exceeds the storage capacity of the buffer and some of the data .overflows. into another buffer, one that the data was not intended to go into (Walfish, Vutukuru, Balakrishnan, Karger, & Shenker, 2006). Since buffers can only hold a specific amount of data, when that capacity has been reached the data has to flow somewhere else, typically into another buffer, which can corrupt data that is already contained in that buffer. Malicious hackers can launch buffer overflow attacks wherein data with instructions to corrupt a system are purposely written into a file in full knowledge that the data will overflow a buffer and release the instructions into the computer's instructions.

4.1.1.2 Ping of death

A type of DoS attack in which the attacker sends a ping request that is larger than 65,536 bytes, which is the maximum size that IP allows. While a ping larger than 65,536 bytes is too large to fit in one packet that can be transmitted, TCP/IP allows a packet to be fragmented, essentially splitting the packet into smaller segments that are eventually reassembled. Attacks took advantage of this flaw by fragmenting packets that when received would total more than the allowed number of bytes and would effectively cause a buffer overload on the operating system at the receiving end, crashing the system (Harris & Hunt, 1999).

4.1.1.3 Smurf Attack.

A type of network security breach in which a network connected to the Internet is swamped with replies to ICMP echo (PING) requests. A smurf attacker sends PING requests to an Internet broadcast address. These are special addresses that broadcast all received messages to the hosts connected to the subnet. Each broadcast address can support up to 255 hosts, so a single PING request can be multiplied 255 times. The return address of the request itself is spoofed to be the address of the attacker's victim. All the hosts receiving the PING request reply to this victim's address instead of the real sender's address (Harris & Hunt, 1999; Mell, Marks, & McLarnon, 2000). A single attacker sending hundreds or thousands of these PING messages per second can fill the victim's high speed internet link with ping replies, bring the entire Internet service to its knees.

4.1.1.4 TCP SYN Attack.

In a SYN attack, a sender transmits a volume of connections that cannot be completed. This causes the connection queues to fill up, thereby denying service to legitimate TCP users (Ohsita, Ata, & Murata, 2006). The TCP SYN Flooding attack takes advantage of the way the TCP protocol establishes a new connection. The attacker generates spoofed packets that appear to be valid new connections to the server. A good solution to the syn attack problem is to modify the TCP implementation to reduce the amount of information stored for each in-progress connection (Harris & Hunt, 1999; Yim & Jung, 2006).

4.1.1.5 Distributed Denial of Service Attack (DDoS).

In and around early 2001 a new type of DoS attack became rampant, called a Distributed Denial of Service attack, or DDoS (Kumar, 2005). In this case multiple comprised systems are used to attack a single target. The flood of incoming traffic to the target will usually force it to shut down. Like a DoS attack, In a DDoS attack the legitimate requests to the affected system are denied (Jeong et al., 2006). Since a DDoS attack it launched from multiple sources, it is often more difficult to detect and block than a DoS attack.

4.1.2 Preventative Measures.

To prevent a system and or a network from becoming a victim of DoS attacks, many preventative solutions, as has been discussed by the various literatures we used above, can be used by administrators to mitigate this. Some of them are as follows:

- Implement router access lists as they will reduce a network's exposure to certain denial-of-service attacks.
- Install patches as they are made available to guard against TCP SYN flooding.
- Disable any unused or unneeded network services. This can limit the ability of an intruder to take advantage of those services to execute a denial-of-service attack.
- Observe your system performance and establish baselines for ordinary activity. Use the baseline to gauge unusual levels of disk activity, CPU usage, or network traffic.
- Routinely examine your physical security with respect to your current needs.
- Invest in and maintain good backups that can be placed into service quickly in the event that a similar primary worker is disabled.
- Invest in redundant and fault-tolerant network configurations.

- Establish and maintain regular backup schedules and policies, particularly for important configuration information.
- Establish and maintain appropriate password policies, especially access to highly privileged accounts such as UNIX root or Microsoft Windows NT Administrator.

4.1.2.1 Antiviral Utilities

In addition to threats from hackers, there is a constant threat of catastrophic loss of data from viruses attached to documents from outside sources, even those from trusted collaborators. The risk of virus infection can be minimized by installing virus-scanning software on servers and locally on workstations. The downside to this often-unavoidable precaution is decreased performance of the computers running antiviral programs, as well as the maintenance of the virus-detection software to insure that the latest virus definitions are installed.

4.1.2.2 Authentication

The most often used method of securing access to a network is to verify that users are who they say they are. However, simple username and password protection at the firewall and server levels can be defeated by someone who either can guess or otherwise has access to the username and password information. A more secure option is to use a synchronized, pseudorandom number generator for passwords. In this scheme, two identical pseudorandom number generators, one running on a credit card–sized computer and one running on a secure server, generate identical number sequences that appear to be random to an observer.

The user carries a credit-card sized secure ID card that displays the sequence on an LCD screen. When a user logs in to the computer network, she uses the displayed number sequence for her password, which is compared to the current number generated by a program running the server. If the sequences match, she is allowed access to the server. Otherwise, she is locked out of the network. Because the number displayed on the ID card—and in the server—changes every 30 seconds, the current password doesn't provide a potential intruder with a way in to the system. The major security hole is that a secure ID card can be stolen, which will provide the thief with the password, but not the username.

More sophisticated methods of user authentication involve biometrics, the automated recognition of fingerprint, voice, retina, or facial features. Authentication systems based on these methods aren't completely accurate, however, and there are often false positives (imposters passing as someone else) and false negatives (an authentic user is incorrectly rejected by the system) involved in the process. In addition to errors in recognition, there are often ways of defeating biometrical devices by bypassing the image-processing components of the systems. For example,

fingerprints are converted into a number and letter sequence that serves as the key to gaining access to network assets; anyone who can intercept that sequence and enter it directly into the system can gain access to the network.

A researcher employed by a biotech firm to analyze nucleotide sequences probably has no need to examine the files in a 3D protein visualization system in the laboratory a few doors down from his office. Similarly, payroll, human resources, and other administrative data may be of concern to the CFO, but not to the manager of the microarray laboratory. Authentication provides the information necessary to provide tiered access to networked resources. This access can be controlled at the workstation, the server, and firewall levels to limit access to specific databases, applications, or network databases.

4.2 ARP Attack.

External attacks by hackers, viruses, worms and trojans are permanent threats to any progressive company. What is not widely known, though, is that the major portions of attacks come from within the network. In 2002 KPMG reported that up to 80 % of all intrusions were initiated internally, from inside a company network (Kumar, 2005). Technical ignorance, curiosity and intentional manipulation of data often lead to serious damages for organizations.

Internal network attacks are typically operated via what is known as ARP Spoofing or ARP Poisoning attacks. Malicious software to run ARP Spoofing attacks can be downloaded on the Internet by anyone (R. Chen, Park, & Marchany, 2007) . Using fake ARP messages, as in figure 14, an attacker can divert all communication between two machines with the result that all traffic is exchanged via his PC. By means of such a man-in-the-middle attack the attacker can in particular run Denial of Service (DoS) attacks to do any combination of the following: -

- Intercept data
- Collect passwords
- Manipulate data
- Even tap VoIP phone calls.

These ARP attacks are usually successful even with encrypted connections like SSL, SSH or PPTP (Khan & Zhang, 2007; Roberts, 2007; Wang, Jin, & Shin, 2007). ARP belongs to the OSI data link layer (layer 2). We will lightly touch on the OSI later for the benefit of first timers.

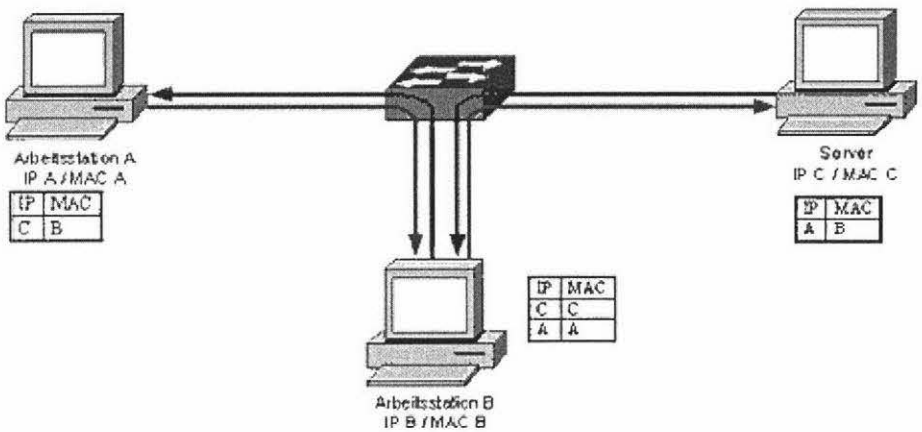


Figure 14: ARP Attack

Some of the symptoms of ARP spoofing attacks are:

- Confidential company information has leaked out and nobody knows how it could have happened.
- Your employees report about intrusion into their online bank account or into their email account.
- Inexplicable incidents have compromised the data of applications that can only be accessed via "secure" web interfaces.
- Strange occurrences in the address resolution (ARP) tables of your network.

ARP Guard is a system that forms an active protection shield against internal ARP attacks(Wang et al., 2007). The ARP Guard early warning system constantly analyzes all ARP messages, sends out appropriate alerts in real-time and identifies the source of the attack. This protection system from ARP spoofing attacks can be configured not only to raise an alarm, if an ARP attack has been sensed, but even to automatically defend against the attacker. ARP Guard easily integrates with most existing IT security environments, such as firewalls, virus scanners, or intrusion detection systems, and forms an active and reliable shield against ARP spoofing.

4.3 The OSI Model

The Open System Interconnection popularly known as the OSI model defines a networking framework for implementing protocols in seven layers, as shown in figure15. Control is passed from one layer to the next, starting at the application layer in one station, proceeding to the bottom layer, over the channel to the next station and back up the hierarchy. Functions at each layer will be briefly discussed below. Further information on this can be found in any computer networking or telecommunications books. The reason we introduce this briefly is to give a first time reader background

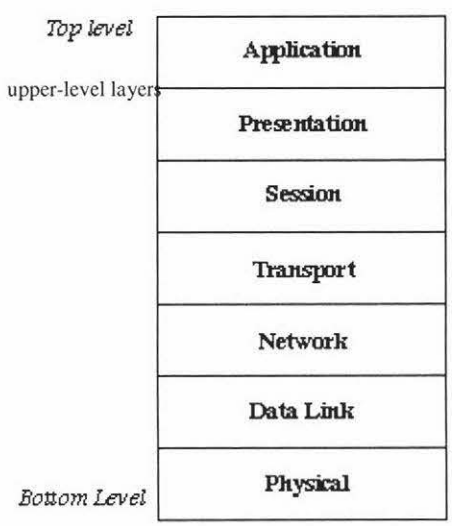


Figure 15: The OSI 7 Layer Model

Physical (Layer 1)

This layer conveys the bit stream - electrical impulse, light or radio signal - through the network at the electrical and mechanical level. It provides the hardware means of sending and receiving data on a carrier, including defining cables, cards and physical aspects. Fast Ethernet, RS232, and ATM are protocols with physical layer components.

Data Link (Layer 2)

At this layer, data packets are encoded and decoded into bits. It furnishes transmission protocol knowledge and management and handles errors in the physical layer, flow control and

frame synchronization. The data link layer is divided into two sublayers: The Media Access Control (MAC) layer and the Logical Link Control (LLC) layer. The MAC sublayer controls how a computer on the network gains access to the data and permission to transmit it. The LLC layer controls frame synchronization, flow control and error checking.

Network
(Layer 3)

This layer provides switching and routing technologies, creating logical paths, known as virtual circuits, for transmitting data from node to node. Routing and forwarding are functions of this layer, as well as addressing, internetworking, error handling, congestion control and packet sequencing.

Transport
(Layer 4)

This chapter discusses the implementation of the Intelligent Mirroring System (IMS) module that can be used a real life working environment. We will analyze query costs that arise from accessin This layer provides transparent transfer of data between end systems, or hosts, and is responsible for end-to-end error recovery and flow control. It ensures complete data transfer.

Session
(Layer 5)

This layer establishes, manages and terminates connections between applications. The session layer sets up, coordinates, and terminates conversations, exchanges, and dialogues between the applications at each end. It deals with session and connection coordination.

Presentation
(Layer 6)

This layer provides independence from differences in data representation (e.g., encryption) by translating from application to network format, and vice versa. The presentation layer works to transform data into the form that the application layer can accept. This layer formats and encrypts data to be sent across a network, providing freedom from compatibility problems. It is sometimes called the *syntax layer.*

Application
(Layer 7)

This layer supports application and end-user processes. Communication partners are identified, quality of service is identified, user authentication and privacy are considered, and any constraints on data syntax are identified. Everything at this layer is application-specific. This layer provides application services for file transfers, e-mail, and other network software services. Telnet and FTP are applications that exist entirely in the application level. User data can traverse the internetwork, as shown in Figure 16, created by joining different network technologies without any challenges.

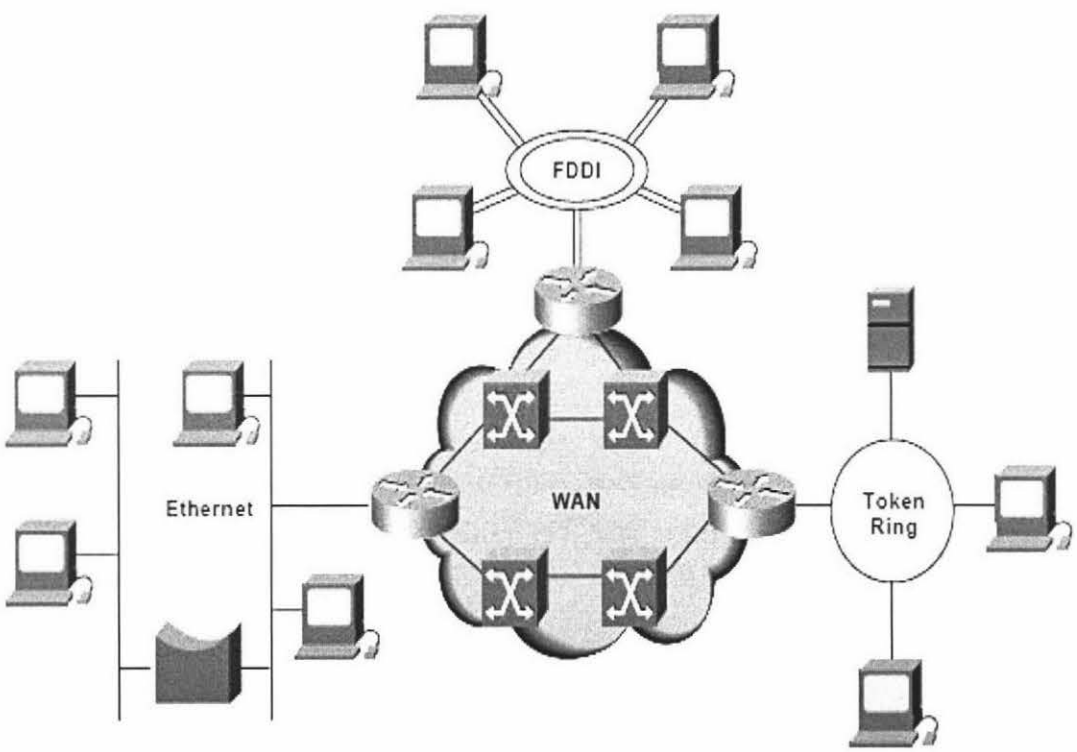


Figure 16: An internetwork created by joining different network technologies

4.3.1 Security at the lower layers – Routers and Firewalls

A firewall is a security device placed at the point of entry between a private network and the outside Internet such that all incoming and outgoing packets have to pass through it. The function

of a firewall is to examine every incoming or outgoing packet and decide whether to accept or discard it. The rules are sequential something like:

- The IP datagram is received.
- The incoming IP datagram is examined to determine if it is destined for a process on the network.
- If the datagram is for the network, it is processed locally.
- If it is not destined for the local network, a search is made of the routing table for an appropriate route and the datagram is forwarded to the appropriate interface or dropped if no route can be found.
- Datagrams from local processes are sent to the routing software for forwarding to the appropriate interface.
- The outgoing IP datagram is examined to determine if there is a valid route for it to take, if not, it is dropped.
- The IP datagram is transmitted.

This function is conventionally specified by a sequence of rules as outlined above, where rules sometimes conflict. To resolve routing conflicts, the decision for each packet is the decision of the first rule that the packet matches. This works in exactly the same way as the router's access list. The current practice of designing a firewall directly as a sequence of rules suffers from three types of major problems:

- the consistency problem, which means that it is difficult to order the rules correctly;
- the completeness problem, which means that it is difficult to ensure thorough consideration for all types of traffic;
- the compactness problem, which means that it is difficult to keep the number of rules small (because some rules may be redundant and some rules may be combined into one rule).

To achieve consistency, completeness, and compactness, a new method was proposed known as structured firewall design, (Gouda & Liu, 2007) which we shall not delve into. An example of a router access list and a firewall basic configuration will be placed in the appendix section of this report to give a basic idea to any interested reader.

4.3.2 **Security at higher layers - Encryption**

At higher levels of the OSI, security is achieved by encrypting data. Encryption is the conversion of data into a form, called a ciphertext, which cannot be easily understood by unauthorized

people. Decryption is the process of converting encrypted data back into its original form, so it can be understood. The topic of data security in such forms as encryption is a subject of study on its own. Books have been written focusing on the topic. Many researchers are proposing ways to help protect bioinformatics data (Fujiyoshi, Imaizumi, & Kiya, 2007; Huang & Liou, 2007).

The use of encryption/decryption is as old as the art of communication. In wartime, a cipher, most people call a "code," although this is not accurate, can be employed to keep the enemy from obtaining the contents of transmissions. Technically speaking, a code is a means of representing a signal without the intent of keeping it secret; examples are Morse code where a letter is represented by a specific bit pattern and ASCII code. Simple ciphers include the substitution of letters for numbers, the rotation of letters in the alphabet, and the "scrambling" of voice signals by inverting the sideband frequencies. More complex ciphers work according to sophisticated computer algorithms that re-arrange the data bits in digital signals. A company dealing with critical data and that wish to stay ahead of its competitors may wish to use encryption to hide the queries it sends to a database, as well as the results it receives back. It then will decrypt this data once it has safely arrived. The problem with this however is the database should be able to decrypt the incoming queries and encrypt its answers.

Chapter 5

Implementation and Result

This chapter discusses the implementation of the Intelligent Mirroring System (IMS) module that can be used a real life working environment. We will analyze query costs that arise from accessing data servers located in various sites and their comparison with getting data locally. The cost benefits of flexible local data mirrors have given rise to the IMS. The discussion covers how the prototype was designed, and how various other factors played a role in coming up with the IMS. This will involve a description of the various structures and objects that are used by the IMS module. XML will be used for the data transfers and analysis. This is because for bioinformatics, most data handling is in XML (Barillot & Archard, 2000; Bruhn & Burton, 2003; De Meo et al., 2006).

5.1 Development Language

The IMS is implemented in Java. The reason for this has mainly been that Java is both platform and network independent. Java programs can run on any system as long as the Java Virtual machine is running on that particular system. This is because Java programs are compiled to an architecture neutral byte code format. For distributed systems such as IMS, this is of utmost importance because clients will be running simultaneously at various multiple sites on different platforms. Using Java, only one version has to be produced to take care of all varying platforms for as long as they support JVM, which greatly simplifies things. Above all, Java comes with an extensive set of classes arranged in packages, which are all readily available to the programmer (Bull, Smith, Ball, Pottage, & Freeman, 2003). Any of these classes can be loaded into a running JVM at any time as Java is a dynamic language. Java is supported by various classes that simplify remote networking and client side interfacing with enterprise servers. The fact that there are many other programmers out there, who can understand and write code in Java, has also contributed to it being chosen as the language of choice for this project. This way, many people can, later on, participate in the further developing of the IMS module. With distributed systems,

security is of utmost importance. One of the most important aspects of Java is the security model (Paul & Evans, 2006). This comprehensive security model greatly simplifies the task of enforcing a strict security policy in a Java application. This security assurance implies that Java applications can download and dynamically load code without having to worry about the possible security implementations (Luo, Heilili, Xu, Zhao, & Lin, 2006).

5.2 Communication Module implementation.

The Intelligent mirroring System is based entirely on the client-server model with the client initiating all the communication. The server will do all the background calculations and only when it is necessary will it call a respective server and upon confirmation, it will do a data dump. The main reason for doing all this is so that we can have a local copy of important data at a local server. This made it necessary for us to use Java Remote Method Invocation (Java RMI) .In all this, we will take both the client interface and the remote interface as clients.

Java RMI enables the programmer to create distributed Java technology-based to Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. It is an object model for developing distributed applications in Java (Sharp & Rountev, 2006). We will be using RMI, objects in one in our clients Java virtual machine (JVM) to invoke methods on objects in the server JVMs. RMI provides powerful features such as object references that cross JVM boundaries, remote invocations that can use entire object graphs as parameters, and distributed garbage collection. RMI can be used either as a stand-alone middleware platform, or a foundation for more advanced architectures.

5.2.1 Logical Architecture

The proposed logically architectures will appear as shown in fig 17

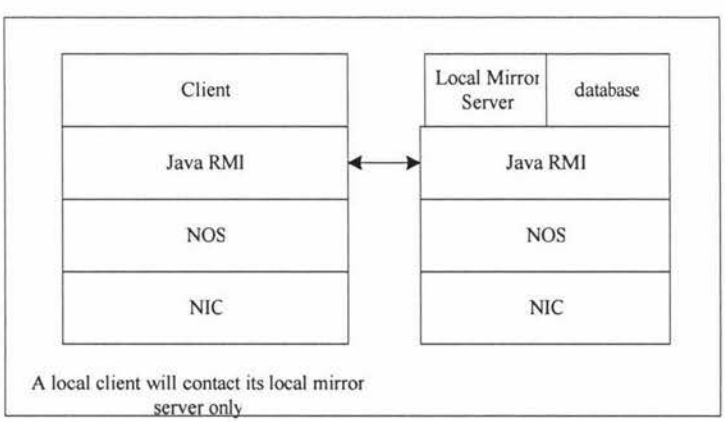


Figure 17: Local communications logical architecture.

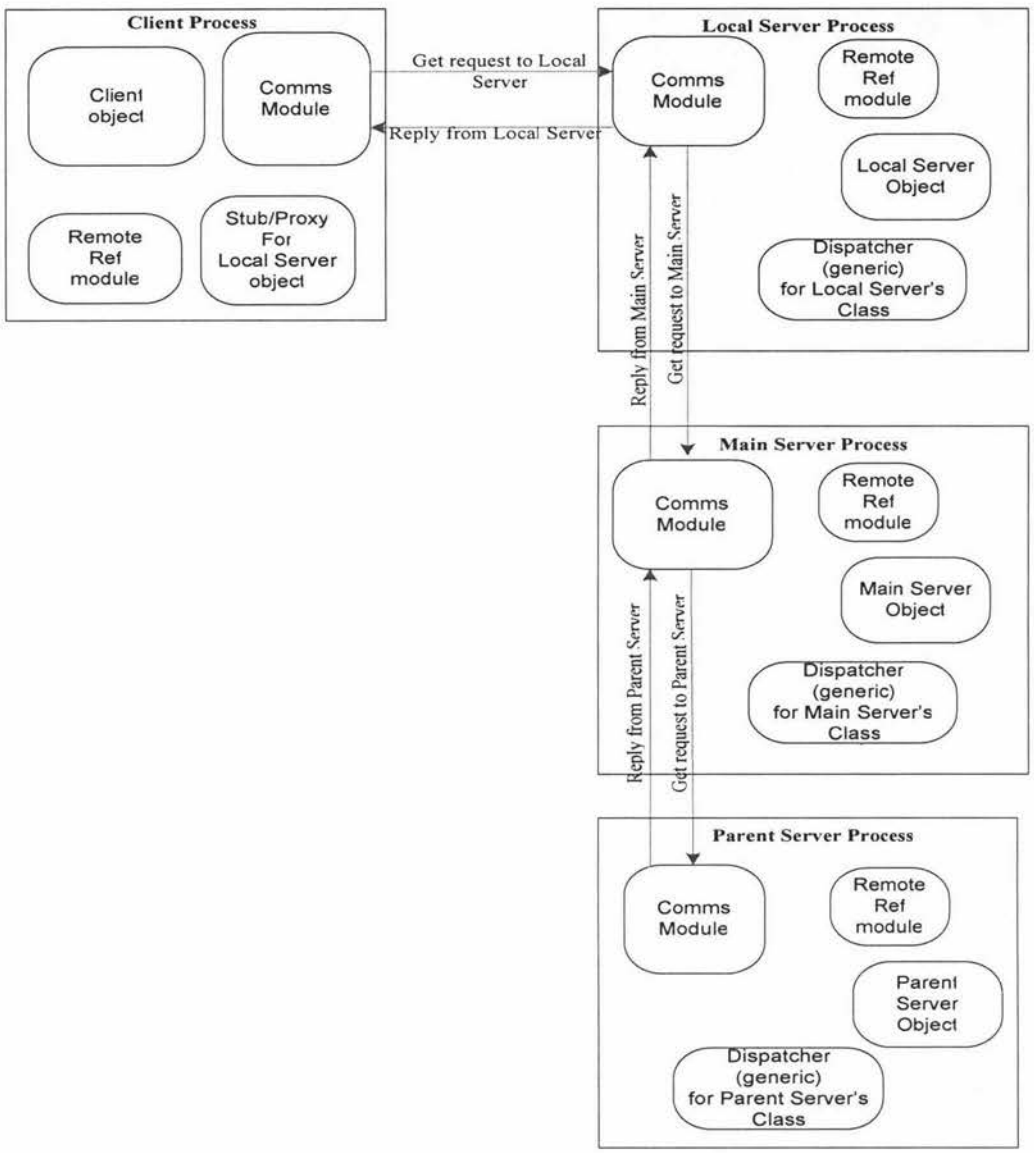


Figure 18: overall communication logical architecture.

To make our model produce the desired results, we will assume that a particular client in a specific country for instance, New Zealand, will strictly consult their local Mirror site only. The logical architecture of figure 18 shows this. Should this site find the information required, it will pass a copy to the requesting client. In the case that it doesn't have this information in its own database, it will forward issue a request to the neighboring server to which it is directly connected. The local server knows about only this server. If this server does have the requested information, it will supply it to the local server which in turn forwards this to the client. Now, assuming it doesn't have the information, the main server will issue a request to the Parent Server, which hopefully has the information to send to the client via both the main server and the local server. The whole process will be transparent to the client. To clarify this, we will use a live

example using the New Zealand situation as this report was written in the country. There is a bio-mirror site at the University of Auckland. Data at this site is mirrored every night from the Australian Bio-mirror, which in turn mirrors the US Bio-mirror. The Auckland University site mirrors only part of the Australian site. This is logical because the people in New Zealand might only be interested in specific data relating say to animal species found only in the country. This is due to both economic and resources reasons. The mirror site in Australia will contain all the information in the New Zealand site as well as additional information. Then, the parent site in the United States of America will be say, part of the INSDC that we mentioned in chapter 4. The parent site will be containing almost all the codes known to man.

5.2.2 Physical Architecture

The proposed prototype currently runs on a single machine. Each process runs in its own copy of the Java Virtual Machine. However, it is easy to distribute this across multiple machines connected via a TCP/IP network. The "localhost" reference used to locate the RMI registry will have to be updated to reflect the new names where the processes are running. The figure below shows a basic representation of the physical architecture of the solution as it currently is, running on a single machine. Client 1 however, is the Administrator (not implemented) that was only used for testing purposes in our case. In real life, we will need this Administrator, no questions, at all sites. Just for coding purpose, this has not been done. In reality, it is a client process with much more rights as the name implies. The advantage with this type of architecture is: the resource server has to communicate with only a few application servers as opposed to many clients. Also, the client process only communicates with an application server via a single middleware interface. Figure 19 shows an n-Tier Middleware Physical architecture (Johnson, 2001).

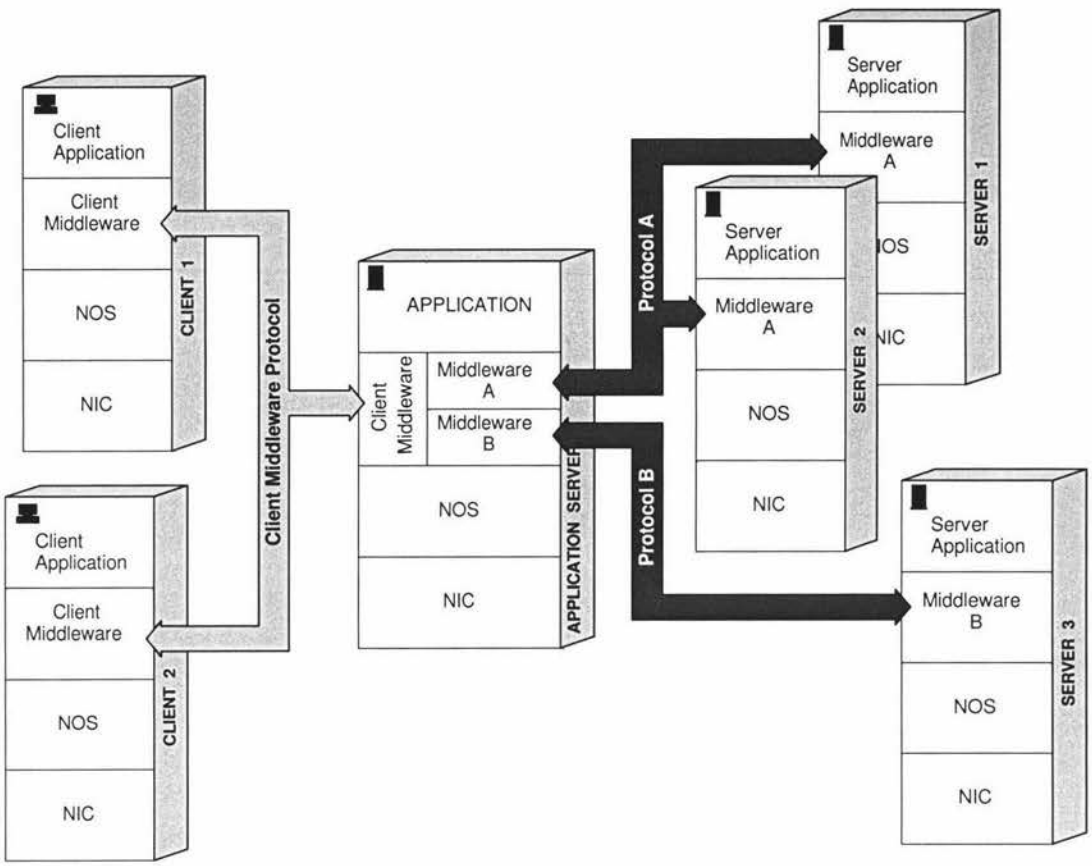


Figure 19: n-Tier Middleware Physical architecture.

5.3 Design Decisions

5.3.1 Cleaning databases.

As we showed earlier, the advancement in the biotechnology field is producing an enormous amount of raw biological data which is accumulating at an exponential rate. Similarly in designing the IMS, it has been noted that we could experience errors, redundancy and discrepancies in the raw data. Although this section has not been implemented, we acknowledge that it is necessary for biological data cleaning. Much work, by other researchers, is in progress to meet this vital goal. Work by Lin et.al. examines the extent of redundancy in biological data and proposes a method for detecting duplicates in this type of data (Lin, Zhu, & Zhang, 2006). Although this suits the Parent Server Database, aspects of it can apply to a local Mirror Server like in our case. Duplicate relations, as shown in figure 20 below, occur mainly in the major databases like Genbank and Swiss-prot. In our prototype, we have assumed harmony of the main centers. We shall remain concerned with what happens after data leaves one of these centers on its way to the final resting place the Local Mirror site.

source	PA2N_VIPAA, accession P34180;	
Organism	Vipera ammodytes ammodytes	Vipera ammodytes ammodytes
Sequence	MRTLWIVAVCLIGVE GNLYQFGNMIFKMTK KSALLSYSNYGCYCG WGGKGKPQDATDRC CFVHDCCYGRVNGC DPKLSIYSYSFENGDI VCGGDDPCLRAVCEC DRVAAICFGENLNTY DKKYKNYPSSHCTET EQC	MRTLWIVAVCLIGVE GNLYQFGNMIFKMTK KSALLSYSNYGCYCG WGGKGKPQDATDRC CFVHDCCYGRVNGC DPKLSIYSYSFENGDI VCGGDDPCLRAVCEC DRVAAICFGENLNTY DKKYKNYPSSHCTET EQC

Figure 20: Duplicate protein records. Record 1 and 2 are protein sequences from different databases.

This kind of error can sip down to our Local mirror and waste valuable space. Work by other researchers has resulted in the accurate identification, to up to 96.8%, of the duplicates in the datasets. Another way of getting rid of such challenges in the prototype would be to implement it with globally unique record identifiers to avoid the situation where there could be two records in the same database with the same unique identifier. This has been considered but since there are already some ways out there already, was not followed. For now, we have to make do with manual management of our prototype. This here is a subject to be explored in future, we believe, with enough time; even better methods can be found especially one to suit the IMS.

5.3.2 Database log.

A log has been implemented in each server process to record server accesses to records. Log records are maintained for each record, detailing the last server which accessed the record and the number of recurring accesses made by the server in a row. The log is maintained in order to guide record migration from server to server. This should always show that lower servers request data from higher servers in the hierarchy and never vice-versa.

5.4 Source code and data

A large part of the work behind this thesis involved writing the source code for the prototype described. This has been a good eye opener to the author who happens to have a strong Computer Networking background and not much software experience. The author's overwhelming desire, to learn new skills was the driving factor behind this. Rather than including the code in this thesis however, we decided to make it available from the Department of Information Systems via a compact disk. Subsequently, it will be available for download on the Internet. The various parts of the code and their state of development are described in the appendix. The appendix also contains some information on the availability of the data sets used in the testing of IMS.

If you have any questions about the source code or data sets then please contact me via email at doug_zw@yahoo.co.nz as I am a bit nomadic.

5.5 RMI Application Overview.

Server processes take two arguments on start-up: their name, with which they bind themselves in the RMI Registry, and the name of the database file to connect to. Where a client cannot connect to its server an error message is displayed. Upon a successful connection, a success message is displayed.

Client process interacts with their servers through the functionality exposed by each server's client interface. When a client wishes to load a record from the database, the appropriate call is made to the server. If the Local Mirror Server cannot find the record in the local database then a connection is made to a higher adjacent Main Server in the hierarchy (provided it has not been made already by another user's initiation). The main server is then asked to interrogate its local database and return the results of its search. The results of these searches are packaged into a container together with appropriate messages and returned to the requesting local server. The client then displays the record (or an error message in the event that the record was not found), and any accompanying messages from the server. If the main server didn't find the requested record, it will forward this request to the Parent Server.

When a local administrator client wishes to delete a record from the database, the appropriate call is made to the server. If the server cannot find the record in the local database no action is taken. If it has the record, this will be deleted. Local administrators have full right to their databases. For now, this will be done manually.

New records require unique identifiers. These are inherited intact from the Parent server. Administrators should not be able to change these as it would result in chaos in real life situation. These identifiers are what is primarily used to check similarities. Sometimes case is sensitive. When an administrator wishes to save a new record to the database, the appropriate call is made to the server. The record is saved and an appropriate message is returned to the administrator for record purposes.

When a new record is created an accompanying log record is also created. Each time a record is accessed its log record is updated with the name of the last accessing server. Where a record is accessed by the same main server a predetermined number of times in a row, that record is migrated (written to the remote main server and a record kept at the parent server). The record at the main server is the only one that can be modified. Any modifications however should be communicated down the hierarchy. This hasn't been implemented.

For now, as log records are maintained in main memory, the log is lost each time the servers are shut down. Thus, whenever a record is requested, the log is examined for an accompanying log record. If one is not found, a new log record is created.

5.6 RMI Application Compilation Instructions.

A single runtime configuration is provided. This should be enough to run on any machine. However, help is readily available should for any reason it fails to run on any machine. The contact email address is doug_zw@yahoo.co.nz. On the compact disk provided is a folder named 899_IMS_Project.

- Start each of the three provided servers in turn from the list of configurations under Run | Run Project.

- Start the client after you have started the servers use Run | Run Project.

The default codebase filepath is:

-Djava.rmi.server.codebase=file:C:\899_IMS_Project\classes\ -Djava.security.policy=file:C:\899_IMS_Project\rmi.policy

This is set already and for as long as you install this project to run off you C: drive, there should not be any problems. It is not necessary to modify the default VM parameters supplied with the configurations. The files used by each server process to store records will be created by the server processes as they start up.

The testing itself will be achieved by using input from plain text files which contains a combination of these commands.

connect servername eg *Local_Mirror //for connecting to a desired servername*

get 0101 *//to retrive and display data for a particular record.*

add 0101, Protein1, Accession1, p1<Description> *//performed by administrator to add record*

update 0201, Protein2, Accession2, p2<Description> *//also by administrator*

delete Main_Mirror *//also by administrator*

These text files are used in place of XML. Basically, wat we are doing here is to assume this is XML data. This assumption has been due to time constraints.

These input files test various scenarios. The first (input 0) test basic commands that an end client can do, basically this is reading from the local database. We haven't been too tight on what the client can do since in this implementation, he is also the administrator. The second focuses on consecutive access after getting a record from an adjacent server five times, it moves to a local server leaving a record from where it came from. This here is the subject of this authors' future work. Chapter 6 will detail this.

Chapter 6

Future Work

The author of this report will be the first one to admit that this was certainly a tip of the iceberg in terms of the study direction. Due to time restrictions, aspects of the proposed IMS system were discussed but not implemented. In future, more programming work needs to be done to cover the aspects we will describe here.

6.1 Database Cleaning.

Data cleaning, also called data cleansing or scrubbing, deals with detecting and removing errors and inconsistencies from data in order to improve the its quality (Galhardas, 2006). Data quality problems are present in single data collections, such as files and databases, for example, due to typing errors during data entry, missing information or other invalid data. When multiple data sources need to be integrated, for example, in data warehouses or biological database systems, the need for data cleaning increases significantly (Elmagarmid, Ipeirotis, & Verykios, 2007; Galhardas, 2006). This is because the sources often contain redundant data in different representations. In order to provide access to accurate and consistent data, consolidation of different data representations and elimination of duplicate information become necessary. Depending on the initial queries from the clients and the nature of the data in the parent server's database, we can have errors. Not much effort was focused towards this direction as it is a strong field of research in its own right. However, in future, focus is needed so that we can at least customise the solution to suit the IMS more. As we have discussed before, all unnecessary data in the IMS has to be cleaned up manually for now mainly because of the size of our database. It remains a fact that data cleaning for semi-structured data like the targeted bioinformatics data, based on XML, is of great importance given the reduced structural constraints and the rapidly increasing amount of XML data (Galhardas, 2006).

As a forward reminder to self or other interested parties, we will touch on requirements that need to be satisfied by a successful data cleaning approach. First of all, it should detect and remove all major errors and inconsistencies both in individual data sources and when integrating multiple sources. The approach should be supported by tools to limit manual inspection and programming effort and be extensible to easily cover additional sources (Rahm, 2000). Rahm went on further to explain that data cleaning should not be performed in isolation but together with schema-related data transformations based on comprehensive metadata. Mapping functions for data cleaning and other data transformations should be specified in a declarative way and be reusable for other data sources as well as for query processing (Rahm, 2000). Especially for data warehouses, a workflow infrastructure should be supported to execute all data transformation steps for multiple sources and large data sets in a reliable and efficient way. It is true that a large body of research seems to deals with schema translation and schema integration while data cleaning is received only little attention in the community. A number of authors in this field seem to focus on the problem of duplicate identification and elimination (Hernandez & Stolfo, 1998; Kim et al., 2003). The keynote paper by Rahm is useful literature in the cleansing regard.

6.2 Implement data modifications down hierarchy.

The current design of the IMS, because of its size and that of the test data, has a major aspect that has not been developed. Future work in this are is critical in order to make this handle real life data as presented by the bioinformatics field. When data is requested from higher servers in the hierarchy, it should only be mirrored from top down. This implies that part of all the data higher up the hierarchy should be mirrored by the server immediately down it. The server that is at the bottom should not contain data that can not be found in the main mirror server. It only mirrors the main server which in turn mirrors the parent server.

The fact that in reality, the middle server also serves some other end mirror servers implies that it can contain much more data than is relevant for a particular end mirror, in this case our local-mirror server. Should this end mirror server require information that is already contained in the main mirror server, then, the main mirror will provide the service without asking the parent server for help. As is, this has been achieved without problems.

The challenge will only come when we want to do this on a much larger scale. Algorithms need to be implemented to achieve it. It is hoped that more work relating to this will come in the future when we take this research further to a higher degree of study possibly a doctorate.

6.3 Queries weighting.

As we have explained before, the IMS is working on a basis that if a client is interested in any data, they request this directly from the server they are connected to. If the end mirror has the data requested, the buck stops there. It will simply supply this data. Now, if it doesn't have the data, it forwards the request upwards and never sideways until it finds what it is looking for. If the requested data is not found, an error is returned to notify the end user. After a number of requests have been made from related or unrelated users connected to an end local mirror, and the mirror doesn't have this information but parent mirrors has, this data is moved for convenience purposes. Now, this is the of most important to the final end product of the IMS. As it is currently, we are using a counting algorithm to achieve this. However, this is not the best way to approach this as the counting algorithm does just that.... COUNT. A more efficient algorithm would be one that takes into account various factors like query cost. The query weighting algorithm itself is a research topic. The writer wishes to explore this, in detail in future. Figure 21 is only to stimulate interest among other researchers.

6.3.1 Query cost.

Not all queries should cost the same value. It should be acceptable to have weightless queries to be allowed to go the whole distance to the parent server as often as predetermined. This could use a count. These are the queries that are deemed by the administrators not to cause much bandwidth utilization when executed. Then there are some moderate queries that will consume considerable bandwidth. These queries will have a higher weighting and will initiate movement of data across to the requesting server in a shorter time that the initial one. The final would be massive data that will almost consume all our bandwidth. This is that data that the administrator would have pinpointed initially. This data, depending on the size and demand, should be moved to the end user sooner rather than later.

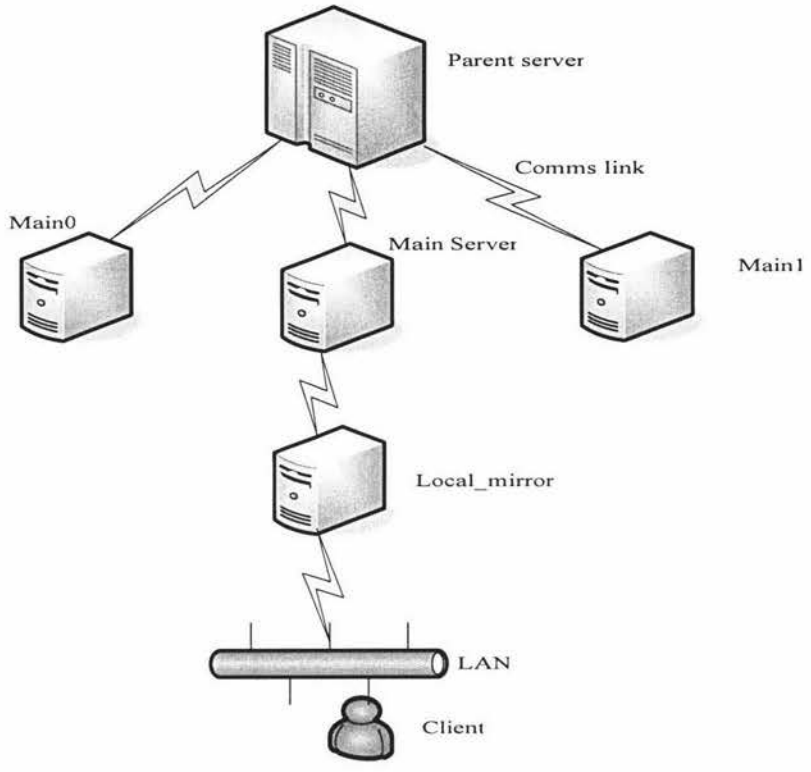


Figure 21: Queries weighting overview.

Chapter 7

Conclusion

This thesis primary goal was to focus on data movements between biological mirror sites. The objectives reflected this by aiming to develop an Intelligent Mirroring System (IMS) that is able to simulate movements of data between various distributed databases. The literature that was explored and investigation of research in the fields of bioinformatics and distributed systems, allowed (with some confidence) the following hypotheses of Section 4.1 to be formed:

1. The movement of data between various mirror sites can be monitored with success using control parameters.
2. Through the addition of additional parameters:
    a) Data saturation at mirror sites can be avoided by automatics checks that will inform responsible authorities.
    b) Information will be communicated managed more effectively by the Intelligent Mirroring System.

The literature review also helped to both identify and provide possible solutions to the ongoing bio-mirroring subproblems stated in Section 3, namely the need to accurately store and move data freely within the wide network among others.

Throughout the entire design and implementation phase of the Intelligent Mirroring System module, the literature was a solid basis upon which design decisions were made.

There is undoubtedly much work yet to be done in the field of bioinformatics data mirroring and accurate storage with Chapter 6 identifying a number of key areas which should be investigated. These are only some of them; one can come up with even more. The main constraint of the project, due to time restrictions, was the inability to implement the algorithm to do accurate data movement calculations to help the administrators to see exactly how the system behaves. As I stated earlier, I have decided to leave this open for future work. Albeit these limitations however,

this thesis has been able to demonstrate a relatively new direction of research. It is through this demonstration that the significance and value of this project is seen to have been confirmed.

Bibliography

Abbas, A. E., & Holmes, S. R. (2004). Bioinformatics and management science: Some common tools and techniques. *Operations Research, 52*(2), 165-190.

Achard, F., Vaysseix, G., & Barillot, E. (2001). XML, bioinformatics and data integration. *Bioinformatics, 17*(2), 115-125.

Arredondo, T., Seeger, M., Dombrovskaia, L., Avarias, J., Calderon, F., Candel, D., et al. (2006). Bioinformatics integration framework for metabolic pathway data-mining. In *Advances in Applied Artificial Intelligence, Proceedings* (Vol. 4031, pp. 917-926).

Ayres, P. E., Sun, H. Z., Chao, H. J., & Lau, W. C. (2006). ALPi: A DDoS defense system for high-speed networks. *Ieee Journal on Selected Areas in Communications, 24*(10), 1864-1876.

Barillot, E., & Archard, F. (2000). XML: a lingua franca for science? *Trends in Biotechnology, 18*(8), 331-333.

Bar-Or, A., Keren, D., Schuster, A., & Wolff, R. (2005). Hierarchical decision tree induction in distributed genomic databases. *Ieee Transactions on Knowledge and Data Engineering, 17*(8), 1138-1151.

Berman, J. J., & Bhatia, K. (2005). Biomedical data integration: using XML to link clinical and research data sets. *Expert Review of Molecular Diagnostics, 5*(3), 329-336.

Bilofsky, H. S., C. Burks, J. W. Fickett, W. B. Goad, F. I. Lewitter et al.,. (1986). The GenBank genetic sequence databank. *Nucleic Acids Res., 14: 1–4.*

Biomirror Public Access. Retrieved 15 June, 2006, from http://bio-mirror.jp.apan.net/biomirror/docs

Bruhn, R. E., & Burton, P. J. (2003). Designing XML schemas for bioinformatics. *Biotechniques, 34*(6), 1200-+.

Bull, J. M., Smith, L. A., Ball, C., Pottage, L., & Freeman, R. (2003). Benchmarking Java against C and Fortran for scientific applications. *Concurrency and Computation-Practice & Experience, 15*(3-5), 417-430.

Chan, M. C., Chang, E. C., Lu, L. M., & Ngiam, P. S. (2006). Effect of malicious synchronization. In *Applied Cryptography and Network Security, Proceedings* (Vol. 3989, pp. 114-129).

Chang, R. K. C. (2002). Defending against flooding-based distributed denial-of-service attacks: A tutorial. *Ieee Communications Magazine, 40*(10), 42-51.

Chen, R., Park, J. M., & Marchany, R. (2007). A divide-and-conquer strategy for thwarting distributed denial-of-service attacks. *Ieee Transactions on Parallel and Distributed Systems, 18*(5), 577-588.

Chen, Y., & Hwang, K. (2006). Collaborative detection and filtering of shrew DDoS attacks using spectral analysis. *Journal of Parallel and Distributed Computing, 66*(9), 1137-1151.

Chen, Y. P. (2005). *Bioinformatics Technologies*: Springer-Verlag Berlin Heidelberg.

Chung, S. Y., & Wong, L. S. (1999). Kleisli: a new tool for data integration in biology. *Trends in Biotechnology, 17*(9), 351-355.

Cochrane, G., Aldebert, P., Althorpe, N., Andersson, M., Baker, W., Baldwin, A., et al. (2006). EMBL Nucleotide Sequence Database: developments in 2005. *Nucleic Acids Research, 34*, D10-D15.

Davenport, G., Ellis, N., Ambrose, M., & Dicks, J. (2004). Using bioinformatics to analyse germplasm collections. *Euphytica, 137*(1), 39-54.

de Knikker, R., Guo, Y. J., Li, J. L., Kwan, A. K. H., Yip, K. Y., Cheung, D. W., et al. (2004). A web services choreography scenario for interoperating bioinformatics applications. *Bmc Bioinformatics, 5*.

De Meo, P., Quattrone, G., Terracina, G., & Ursino, D. (2006). Integration of XML Schemas at various "severity" levels. *Information Systems, 31*(6), 397-434.

d'Inverno, M., & Prophet, J. (2005). Multidisciplinary investigation into adult stem cell behavior. In *Transactions on Computational Systems Biology Ii* (Vol. 3737, pp. 49-64).

Elmagarmid, A. K., Ipeirotis, P. G., & Verykios, V. S. (2007). Duplicate record detection: A survey. *Ieee Transactions on Knowledge and Data Engineering, 19*(1), 1-16.

Fujiyoshi, M., Imaizumi, S., & Kiya, H. (2007). Encryption of composite multimedia contents for access control. *Ieice Transactions on Fundamentals of Electronics Communications and Computer Sciences, E90A*(3), 590-596.

Galhardas, H. (2006). Data cleaning and transformation using the AJAX framework. In *Generative and Transformational Techniques in Software Engineering* (Vol. 4143, pp. 327-343).

Gilbert, D., Ugawa, Y., Buchhorn, M., Wee, T. T., Mizushima, A., Kim, H., et al. (2004). Bio-Mirror project for public bio-data distribution. *Bioinformatics, 20*(17), 3238-3240.

Gouda, M. G., & Liu, A. X. (2007). Structured firewall design. *Computer Networks, 51*(4), 1106-1120.

Harris, B., & Hunt, R. (1999). TCP IP security threats and attack methods. *Computer Communications, 22*(10), 885-897.

Hernandez, M. A., & Stolfo, S. J. (1998). Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Mining and Knowledge Discovery, 2*(1), 9-37.

Hoschek, W., Jaen-Martinez, J., Samar, A., Stockinger, H., & Stockinger, K. (2001). Data management in an International Data Grid project. In *Grid Computing - Grid 2000, Proceedings* (Vol. 1971, pp. 77-90).

Huang, E. W., & Liou, D. M. (2007). Performance analysis of a medical record exchanges model. *Ieee Transactions on Information Technology in Biomedicine, 11*(2), 153-160.

Jeong, S., Kim, H., & Kim, S. (2006). An effective DDoS attack detection and packet-filtering scheme. *Ieice Transactions on Communications, E89B*(7), 2033-2042.

Johnson, E. J. (2001). *The complete guide to Client/Server computing.* (1st ed.): Prentice Hall.

Khan, M. K., & Zhang, J. S. (2007). Improving the security of 'a flexible biometrics remote user authentication scheme'. *Computer Standards & Interfaces, 29*(1), 82-85.

Kim, W., Choi, B. J., Hong, E. K., Kim, S. K., & Lee, D. (2003). A taxonomy of dirty data. *Data Mining and Knowledge Discovery, 7*(1), 81-99.

King, G. J. (2004). Bioinformatics: harvesting information for plant and crop science. *Seminars in Cell & Developmental Biology, 15*(6), 721-731.

Kumar, S. (2005). Impact of Distributed Denial of Service (DDoS) attack due to ARP storm'. In *Networking - Icn 2005, Pt 2* (Vol. 3421, pp. 997-1002).

Lacroix, Z., Critchlow, T. (2003). *Bioinformatics: Managing Scientific Data*: Morgan Kaufmann Publishers.

Lin, K., Zhu, L., & Zhang, D. Y. (2006). An initial strategy for comparing proteins at the domain architecture level. *Bioinformatics, 22*(17), 2081-2086.

Liu, M. Y., & Grigoriev, A. (2005). Fast parsers for entrez gene. *Bioinformatics, 21*(14), 3189-3190.

Luo, Z. X., Heilili, N., Xu, D. W., Zhao, C., & Lin, Z. Q. (2006). Web application security gateway with Java non-blocking IO. In *Next Generation Information Technologies and Systems, Proceedings* (Vol. 4032, pp. 96-105).

Masys, D. R. (1989). New Directions in Bioinformatics. *Journal of Research of the National Institute of Standards and Technology, 94*(1), 59-63.

Mell, P., Marks, D., & McLarnon, M. (2000). A denial-of-service resistant intrusion detection architecture. *Computer Networks-the International Journal of Computer and Telecommunications Networking, 34*(4), 641-658.

Ohsita, Y., Ata, S., & Murata, M. (2006). Detecting distributed denial-of-service attacks by analyzing TCP SYN packets statistically. *Ieice Transactions on Communications, E89B*(10), 2868-2877.

Okubo, K., Sugawara, H., Gojobori, T., & Tateno, Y. (2006). DDBJ in preparation for overview of research activities behind data submissions. *Nucleic Acids Research, 34*, D6-D9.

Park, J., Lee, C., & Park, J. C. (2005). Information visualization with text data mining for knowledge discovery tools in bioinformatics. In *On the Convergence of Bio-Information-, Environmental-, Energy-, Space- and Nano-Technologies, Pts 1 and 2* (Vol. 277-279, pp. 259-265).

Paul, N., & Evans, D. (2006). Comparing Java and .NET security: Lessons learned and missed. *Computers & Security, 25*(5), 338-350.

Pinheiro, M., Afreixo, V., Moura, G., Freitas, A., Santos, M. A. S., & Oliveira, J. L. (2006). Statistical, computational and visualization methodologies to unveil gene primary structure features. *Methods of Information in Medicine, 45*(2), 163-168.

Rahm, E. D., Hong-Hai. (2000). Data Cleaning: Problems and Current Approaches. *IEEE Bulletin of the Technical Committee on Data Engineering., Vol 23*(No. 4), 11.

Richardson, D. E., Vanwye, J. D., Exum, A. M., Cowen, R. K., & Crawford, D. L. (2007). High-throughput species identification: from DNA isolation to bioinformatics. *Molecular Ecology Notes, 7*(2), 199-207.

Roberts, C. (2007). Biometric attack vectors and defences. *Computers & Security, 26*(1), 14-25.

Shabo, A., Rabinovici-Cohen, S., & Vortman, P. (2006). Revolutionary impact of XML on biomedical information interoperability. *Ibm Systems Journal, 45*(2), 361-372.

Shah, S. P., Huang, Y., Xu, T., Yuen, M. M. S., Ling, J., & Ouellette, B. F. F. (2005). Atlas - a data warehouse for integrative bioinformatics. *Bmc Bioinformatics, 6*.

Sharp, M., & Rountev, A. (2006). Static analysis of object references in RMI-based Java software. *Ieee Transactions on Software Engineering, 32*(9), 664-681.

Strizh, I. G. (2006). Ontologies for data and knowledge sharing in biology: plant ROS signaling as a case study. *Bioessays, 28*(2), 199-210.

Sun, M. G., Shi, Y. Q., Liu, Q., & Sclabassi, R. J. (2005). Data integration for medical information management. *Journal of Vlsi Signal Processing Systems for Signal Image and Video Technology, 41*(3), 319-328.

Thompson, J. D., & Poch, O. (2006). Multiple sequence alignment as a workbench for molecular systems biology. *Current Bioinformatics, 1*(1), 95-104.

Varfolomeev, S. D., Uporov, I. V., & Fedorov, E. V. (2002). Bioinformatics and molecular modeling in chemical enzymology. Active sites of hydrolases. *Biochemistry-Moscow, 67*(10), 1099-1108.

Walfish, M., Vutukuru, M., Balakrishnan, H., Karger, D., & Shenker, S. (2006). DDoS defense by offense. *Computer Communication Review, 36*(4), 303-314.

Wang, H. N., Jin, C., & Shin, K. G. (2007). Defense against spoofed IP traffic using hop-count filtering. *Ieee-Acm Transactions on Networking, 15*(1), 40-53.

Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *Ieee Transactions on Neural Networks, 16*(3), 645-678.

Yim, H. B., & Jung, J. I. (2006). IP traceback algorithm for DoS/DDoS attack. In *Management of Convergence Networks and Services, Proceedings* (Vol. 4238, pp. 558-561).

Appendix

A Sample Firewall Configuration

In section 4.3.1, we briefly discussed the fundamentals of firewall configuration. We will now look at what a firewall configuration might actually look like without adding confusion I hope. This has been adopted from a learning forum the author uses for private studies. This is another of my interest areas of study.

Basically, there are several versions to achieving the same goal and network administarators can follow any they are comfortable with. Some of these are **ipfwadm** command (or the **ipfwadm-wrapper** script), the second is **ipchains**, and the one that uses **iptables**. The iptable one is what we will tackle here as it is easer to explain. There are similarities and differences between these various configuration tool syntaxes:

```
#!/bin/bash
#############################################################################
# IPTABLES VERSION
# This sample configuration is for a single host firewall configuration
# with no services supported by the firewall machine itself.
#############################################################################

# USER CONFIGURABLE SECTION

# The name and location of the ipchains utility.
IPTABLES=iptables

# The path to the ipchains executable.
PATH="/sbin"

# Our internal network address space and its supporting network device.
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"

# The outside address and the network device that supports it.
ANYADDR="0/0"
ANYDEV="eth1"

# The TCP services we wish to allow to pass - "" empty means all ports
```

```
# note: comma separated
TCPIN="smtp,www"
TCPOUT="smtp,www,ftp,ftp-data,irc"

# The UDP services we wish to allow to pass - "" empty means all ports
# note: comma separated
UDPIN="domain"
UDPOUT="domain"

# The ICMP services we wish to allow to pass - "" empty means all types
# ref: /usr/include/netinet/ip_icmp.h for type numbers
# note: comma separated
ICMPIN="0,3,11"
ICMPOUT="8,3,11"

# Logging; uncomment the following line to enable logging of datagrams
# that are blocked by the firewall.
# LOGGING=1

# END USER CONFIGURABLE SECTION
################################################################################
# Flush the Input table rules
$IPTABLES -F FORWARD

# We want to deny incoming access by default.
$IPTABLES -P FORWARD deny

# Drop all datagrams destined for this host received from outside.
$IPTABLES -A INPUT -i $ANYDEV -j DROP

# SPOOFING
# We should not accept any datagrams with a source address matching ours
# from the outside, so we deny them.
$IPTABLES -A FORWARD -s $OURNET -i $ANYDEV -j DROP

# SMURF
# Disallow ICMP to our broadcast address to prevent "Smurf" style attack.
$IPTABLES -A FORWARD -m multiport -p icmp -i $ANYDEV -d $OURNET -j DENY

# We should accept fragments, in iptables we must do this explicitly.
$IPTABLES -A FORWARD -f -j ACCEPT

# TCP
# We will accept all TCP datagrams belonging to an existing connection
# (i.e. having the ACK bit set) for the TCP ports we're allowing through.
# This should catch more than 95 % of all valid TCP packets.
$IPTABLES -A FORWARD -m multiport -p tcp -d $OURNET --dports $TCPIN /
    ! --tcp-flags SYN,ACK ACK -j ACCEPT
$IPTABLES -A FORWARD -m multiport -p tcp -s $OURNET --sports $TCPIN /
    ! --tcp-flags SYN,ACK ACK -j ACCEPT


# TCP - INCOMING CONNECTIONS
# We will accept connection requests from the outside only on the
# allowed TCP ports.
$IPTABLES -A FORWARD -m multiport -p tcp -i $ANYDEV -d $OURNET $TCPIN /
```

```
    --syn -j ACCEPT

# TCP - OUTGOING CONNECTIONS
# We will accept all outgoing tcp connection requests on the allowed /
    TCP ports.
$IPTABLES -A FORWARD -m multiport -p tcp -i $OURDEV -d $ANYADDR /
    --dports $TCPOUT --syn -j ACCEPT
# UDP - INCOMING
# We will allow UDP datagrams in on the allowed ports and back.
$IPTABLES -A FORWARD -m multiport -p udp -i $ANYDEV -d $OURNET /
    --dports $UDPIN -j ACCEPT
$IPTABLES -A FORWARD -m multiport -p udp -i $ANYDEV -s $OURNET /
    --sports $UDPIN -j ACCEPT
# UDP - OUTGOING
# We will allow UDP datagrams out to the allowed ports and back.
$IPTABLES -A FORWARD -m multiport -p udp -i $OURDEV -d $ANYADDR /
    --dports $UDPOUT -j ACCEPT
$IPTABLES -A FORWARD -m multiport -p udp -i $OURDEV -s $ANYADDR /
    --sports $UDPOUT -j ACCEPT
# ICMP - INCOMING
# We will allow ICMP datagrams in of the allowed types.
$IPTABLES -A FORWARD -m multiport -p icmp -i $ANYDEV -d $OURNET /
    --dports $ICMPIN -j ACCEPT
# ICMP - OUTGOING
# We will allow ICMP datagrams out of the allowed types.
$IPTABLES -A FORWARD -m multiport -p icmp -i $OURDEV -d $ANYADDR /
    --dports $ICMPOUT -j ACCEPT
# DEFAULT and LOGGING
# All remaining datagrams fall through to the default
# rule and are dropped. They will be logged if you've
# configured the LOGGING variable above.
#
if [ "$LOGGING" ]
then
        # Log barred TCP
        $IPTABLES -A FORWARD -m tcp -p tcp -j LOG
        # Log barred UDP
        $IPTABLES -A FORWARD -m udp -p udp -j LOG
        # Log barred ICMP
        $IPTABLES -A FORWARD -m udp -p icmp -j LOG
fi
#
# end.
```

This is only a simple example, a person configuring has to be very careful to ensure it does what they want while implementing it.

Filters Using Extended Access Lists

Extended ACLs compare the source and destination addresses of the IP packets to the addresses that are configured in the ACL in order to control traffic. Extended ACLs also provide a means to filter traffic based on specific protocols. This provides a more granular control for the implementation of filters on a network.

Extended ACLs allow a client to access some resources on the network while the client cannot access the other resources. For example, you can implement a filter that allows DHCP and Telnet traffic to the client while it restricts all other traffic.

This is the command syntax of extended ACLs:

```
access-list access-list-number [dynamic dynamic-name [timeout minutes]]
{deny | permit} protocol source source-wildcard
destination destination-wildcard [precedence precedence]
[tos tos] [log | log-input] [time-range time-range-name]
```

Extended ACLs can use numbers in the range of 100 to 199. Extended ACLs can also use numbers in the range of 2000 to 2699. This is the expanded range for extended ACLs. We shall restrict this to only extended ACLs as standard ACls are too restrictive.