Transactions on Information and Communications Technologies vol 16, © 1996 WIT Press, www.witpress.com, ISSN 1743-3517

Artificial intelligence for secure network communications

Julia Johnson(*), Tim Derzaph(**) & Jon Firor(***)
(*) *Department of Computer Science, University of Regina, Regina, SK, S4S 0A2, Canada*
(**) *Human Resource Development Canada, Regina, SK, S4P 2H9, Canada*
(***) *Attachmate Canada Incorporated, British Columbia, V5C 6C6, Canada*
Email: johnson@cs.uregina.ca

Abstract

The national telecommunications infrastructure is increasing rapidly in complexity and the numbers of interconnected networks. To ensure high quality telecommunications service, Dr. Johnson and her students are developing an expert system that monitors evolution of the network and rejects proposals for evolution that violate rules that are built into the system. They are using an object-oriented graphical language for the rules and for constructing the user interface. Initially, they are concerned with changing network topologies and the software developed is called an expert system for network configuration. This is one part of their research goal of providing a network planning system that plans network topologies based on demands for service. Their focus is on interconnected voice networks as opposed to the rapidly emerging data services (e.g., email). The methods are expected to apply also to data networks.

Three students are involved all of whom are now employed full time at telecommunications companies in Canada and continue their work on the project by providing consultation and writing scholarly papers coauthored by Dr. Johnson. We report investigative and preliminary work undertaken by Johnson and her students at the University of Regina in the area of network communications security, modelling and management.

1 Introduction

This paper reports on several projects undertaken at the University of Regina in the area of network communications security and management. The projects are part of a research program funded by the National Sciences and Engineering Research Council of Canada (NSERCC) and Telecommunications Research Laboratories Regina. The projects were directed by Johnson and carried out by her students who are now working in industry and government. Additionally, some follow up work carried out by Johnson's graduate students is described.

Short articles were written by two students reporting investigative work on using artificial intelligence for network modelling, management and security. The papers summarized here were titled as follows:

1. *A first look at network modelling using DEFINE* by Jon Firor, Attachmate Canada Inc., Burnaby, British Columbia, Canada
2. *Combination door lock algorithms* by Tim Derzaph, Human Resource Development Canada, Regina, SK, Canada.

A project *"A Knowledge-based transition to open communication"* (Abouzgaia 1994) funded by Telecommunications Research Labs Regina builds on the above two subprojects.

Firor reports negative results concerning the use of the language DEFINE for network modelling. Firor's work is, nevertheless, useful because he tried to model networks by insisting that their structure parallel the structure of the expression needed to describe them. Abouzgaia, although successful, models networks by data, that is by *applying* the language DEFINE, without the requirement that the DEFINE expression represents the object being modeled. Abouzgaia solves a practical problem in the most straightforward way given the tools at hand (the concepts of the SET model and the language DEFINE). Firor, on the other hand, provides an approach within which it would be possible to construct a theory of network processes. He succeeded in identifying where the current theory of network process (embodied in the language DEFINE) needs refinement.

Derzaph's work (Combination door lock algorithms) serves as an example of how the concepts of the SET model and the language DEFINE provide the primitives to express security in networks.

A third subproject described in this report is to provide network monitor (sniffer based) tools for security in case of accidental deletion of data. This research involves writing application specific software for 1) surveillance and 2) security. Dr. Johnson is actively pursuing industry groups and government agencies to bring the product to market. For example, she has made a proposal to the Director of the Canadian Security Intelligence Service.

2 A First Look at Network Modelling Using DEFINE

The original project proposal attached in the Appendix outlines the strategy and reasoning behind our reasons and methods for network modelling. Although the results were not as positive as we would have desired we did learn a great deal from the process.

The need for automated management is becoming critical as networks grow both in terms of size and complexity. In order to better manage their networks, the telecommunications industry has developed the approach as shown in Figure 1. Standards are in place (CMIP etc.) for the structure of data originating at network elements (NEs), the communication link (T1.214), and the network operating systems. Although these systems are an improvement to the management of networks, they lack the ability to run without operation intervention and lack any real "AI" reasoning capability. The data streams being fed to the OSes (operating systems) are just that, "data streams". The OSes lack any understanding as to what this data is or what it applies to. The OSes can act on certain "alarms" and "triggers", but they may not formulate any solutions on their own and they may not learn.

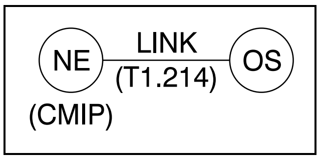

Figure 1

How can the above problems be addressed? The writer would suggest that in order for the OS to make decisions or solutions it must have some learning capabilities and it must also have some understanding as to what it is that it is dealing with. The learning and rule capabilities could be build into an Intelligent Management Engine (IME) that receives data from the OS and passes instructions back to it. This leaves the IME free to act quickly in formulating network management "ideas". The other piece required is a model of the entities and the relationships between them that exist in the network. This was the primary focus of the work conducted during this project.

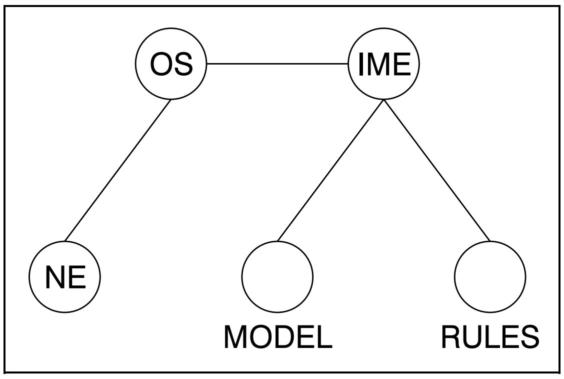

Figure 2

2.1 The Language DEFINE

Natural-deduction-based set theory, NaDSet, was introduced by Gilmore. The G-notes (Gilmore 1987; 1988) describe applications of an earlier form of NaDSet in which increasing levels of abstraction are defined as objects and reasoned about (paraphrased from Gilmore & Tsiknis 1993 p. 255).

We use the earlier version of the logic NaDSet to model communication networks. The language for modelling is called DEFINE. We wish to see where the language breaks down for network modelling to be in a better position to propose a general purpose modelling language with some version of DEFINE as its kernel. DEFINE appears particularly suitable for network modelling because the fundamental component in a network (like in DEFINE) is the 'ordered pair' or 'link' in network terminology.

A set is a collection of objects each possessing a property that distinguishes objects that are members of the set from those that are not. An example of a set is "the network cards in a network." The intension of a set is the property that determines membership in the set. The extension of a set is the membership of the set (the collection of objects that satisfy the intension of the set). The intension of the above set is the property of being a network card in the given network. The extension of the set is the collection of network cards that currently exist in the network.

A 'base set' is a set whose meaning cannot be expressed formally in a computer language such as DEFINE. However, 'defined sets' are declared by expressing their meaning using DEFINE in terms of base sets and previously declared defined sets. A principle for good modelling is: Never introduce a new base set when the set could be declared as a defined set from previously declared base and defined sets.

2.2 A Modelling Attempt

All networks consist of two basic parts: elements and the links that connect them. From these two entities all networks can be constructed (see the Appendix for detail). The definition for a link using DEFINE is as follows:

Link for {Nodes x Nodes x Types | [for some a,b:Nodes] not (a=b) | a node is a link between two nodes that are not the same node and there is a type associated with it}

Nodes are a primitive string identifier but if the model progressed further, a type field would have been included to specify a node's function. A "loose" definition of types is as follows:

Types for {physical x speed x MAC_access_scheme x ...||}

The physical type could be twisted pair, coaxial, fibre, etc. with these types having further attributes. The speed is the speed at which the link operates. There could be fields such as MAC_access_scheme (CSMA, token passing, etc.) and various other types. The primary focus at this point is on the definitions for the various topologies and we will fill in the details later.

Figure 3 shows a point to point network. A point to point network could be defined as follows:

Point_to_point for {nodes x nodes x links||}

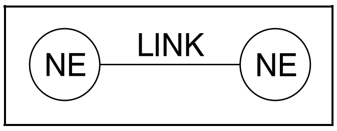

Figure 3

Although point to point networks are the basis for the network model, the definition used here has been left loose and could be refined at a later stage. Point to point networks are rarely seen isolated as shown. They are usually parts of a larger networks such as the bus network shown in Figure 4.

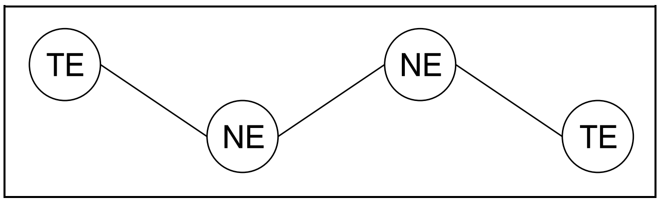

Figure 4

The bus topology, Figure 4, consists of several nodes connected in a linear fashion. A bus topology is really a collection of point to point networks connected in such a way that no element in the bus network is connected to more than two other elements on the bus. All bus networks have a special feature in that they must contain two and only two end elements known as termination elements (TE). These termination elements are connected to one and only one other element. The reason for this will become apparent when we investigate the ring topology. Buses may contain redundancy as previously outlined.

A very crucial definition is now given below for a bus network. This is where the deficiency of DEFINE becomes apparent. Before giving this definition, a definition is given for terminators.

Terminator for { <Node,type>:Nodes | ([for some <Node,nodeA2,typeA>:link] and [for all <nodeB1,nodeB2,typeB>:link] (not ((Node=nodeB1 or Node=nodeB2) and typeA=typeB)) or ([for some <nodeA1,Node,typeA>:link] and [for all <nodeC1,nodeC2,typeC>:link] (not ((Node=nodeC1 or Node=C2) and typeA=typeC)))) |a node can only be a terminator unless it is not the first or the second element in any tuple that is a member of links unless the link types are different}

The above looks rather complex but it is just saying that in order to be a terminator, a node must have only one link of the same type to other nodes. Physically, a terminator is the last node on the bus.

At this point what was needed was a recursive definition for bus networks defined in English as follows:

Bus Network: **Two entities A and B connected together with one link where A of type1 is an element and B is an element or bus network of the same type.**

One attempt of many to define this condition is as follows:

Bus_network for {<nodeA,element,type>:link | [for some c:nodes] <nodeA,c,type2>:link and ((c,nodeB,type3>:link or <c,nodeB,type3>:bus_network) and type=type2=type3) | }

How accurate must the model be? Just as humans do not have to understand quantum physics to understand electricity or computers, a computer may not have to entirely understand the entity(ies) that it is dealing with. With all humans the writer believes that there comes a point where we no longer can justify or explain why we know something. At some point we make the leap of faith and just assume that the fundamentals of our knowledge are sound and it is on these that we build. At some point in our education we accept what we are told and it is on these ideas that we build. That raises the question, "How accurate must the model be in order for a computer to make accurate inferences based upon it?" In particular is it extremely important to have as few base sets as possible in our model? The writer would suggest that in the case of network modelling that MAY not be essential. Computer scientists understand network protocols and communications schemes BUT very few understand the physics behind the transfer of electrons and that is usually not required. Alan Turing once said that at a certain level of conception he believed that "sparks would fly," the machine having achieved true intelligence. The writer believes that the model need not be extremely defining (that is, compromising a high proportion of sets whose intentions can be expressed formally in a language intended to be interpreted by a computer) but more importantly it MUST be unambiguous and logical. According to the G-notes, the above statement for **Bus_network** is disallowed.

What is the importance of these types of self-referential statements being disallowed? The problem with this restriction is that it becomes very difficult if not impossible to define entities such as bus networks of **arbitrary** size.

Why does this not work? The domain graph of a recursive definition best shows the reason for this. Shown in Figure 5 is a recursive domain graph. The problem with it is that it contains a **directed cycle**. This is expressly not allowed in DEFINE. The writer suspects that the reason for this is that cyclic graphs like this make it difficult to ascertain the base sets.

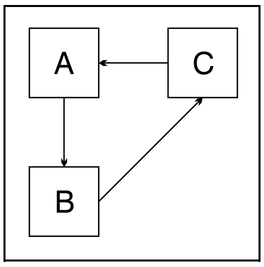

Figure 5

Gilmore himself has reached a similar conclusion. In his words, "... an artificial restriction on abstraction imposed by the second order logic within which [an earlier version of the logic NaDSet] was formulated hindered applications of that logic." (Gilmore & Tsiknis 1993 p. 255)

3 Combination Door Lock Algorithms for Network Security

In this section we argue that the primitives to express security in networks lie in combination door lock algorithms. Section 3.1 presents the algorithms and Section 3.2 applies a more complicated combination lock to communications network security.

3.1 Details of the Security Algorithms

The following are some algorithms that may be used for push button door locks. It is assumed that these locks are of the standard type which may have the combination easily changed, and allow each of the numbered buttons to be pushed only once. In addition these locks can have one or more buttons pushed simultaneously. In any examples presented, individual numbers pushed in sequence will be separated by commas. For example if we are to push the button with the number 1 next to it then after push the button with the number 2 next to it we will write "push 1,2". If these two buttons are to be pushed at the same time, we will write "push 12".

The purpose of proposing these algorithms is to attempt to offer a balance between the extreme cases of regularly changing the combination to some totally random value (which provides the highest possible level of security but is impractical, as most people often forget the combinations, thereby making it an inconvenient solution) and of not changing the combination at all, to provide an easy method by which to remember the values (for example 1,2,3,4,5).

The basic concept is to use a mathematical formula that can be applied to some regularly changing seed value, and which can be easily and quickly computed by any individual without an external aid. Since more digits used decrease the odds of guessing the code and increase the level of security, it is proposed that all buttons be utilized. Of course, if any of the constraints set out here are revealed to non privileged users of the door locks, the system's usefulness will be reduced proportional to the number of constraints revealed. In other words, if the proposal described in this document is to be implemented, its full details should not be revealed to anyone other than the code maker, and only the relevant sections revealed to the door lock users.

To begin considering what type of code to use, it is proposed that the number of doors used be a consideration. For example, if a maximum of 3 doors with different push button locks are allowed in any code system, the first digit could be a door designation value. Suppose 3 doors in a building are to be used, with one door located at the North end of the building, one at the East end of the building, and the last door at the South (or West) end of the building. Start at the North door designating it number 1. Rotating clockwise, the other doors could be numbered 2 for the East door and 3 for the South (West) door. In this case, the door number corresponds to the first digit pushed on any of the doors (unique number in each case). Alternatively, a single common digit can be used on all doors, but varying the order in which it is to be pushed. For example, the first digit of the code on the first door would be the number 1, and the same number would be used but as the second digit of the code on the East door, and so on.

For the remaining buttons to be pushed, a seed and formula need to be agreed upon. One option is to use the month as a seed. So for January one could use 1, for February use 2, and so on until December where 12 could be used (a binary representation utilizing 3 or more digits is required as illustrated in an example to follow). Since the month digit values could overlap with the door position values mentioned earlier, the code scheme to be used must take conflicts like this into consideration.

The following is an example of the principles discussed, implementing them into a code system which could work on a three door system, and allowing the seed to change every four months (corresponding to normal academic semesters). This scheme would improve security while reducing the randomness of an unformulated code system. Assume as before that combination door locks are situated at the North, East, and West ends of the building and that each lock has five numbered buttons.

1) For each four month block (for each normal semester in an academic setting) number the months in binary and press this code value in first, using the last two buttons only (the buttons numbered 4 and 5). A zero indicates no button to push unless both digits are zero, then push both buttons simultaneously.

Therefore:

January	1	2	3	4	5	push 5
				0	1	
February	1	2	3	4	5	push 4
				1	0	
March	1	2	3	4	5	push 4,5
				1	1	
April	1	2	3	4	5	push 45
				0	0	

2) Determine if the current door is at the North, East, or West end of the building. Then in decimal, press the door designation number using the first three buttons only (numbered 1, 2, 3). The star indicates which button is to be pressed.

Therefore:

North door	1	2	3	push 1
	*			
East door	1	2	3	push 2
		*		
West door	1	2	3	push 3
			*	

3) This step will be updated each semester. The seed rule is: "press the remaining unused buttons starting at the lowest numbered button and continuing to the highest numbered button, until all buttons have been used. One button is to be pushed at a time."

Therefore:

For the January to April duration on the North door, the following would apply:

January	5, 1, 2, 3, 4
February	4, 1, 2, 3, 5

March	4, 5, 1, 2, 3
April	45, 1, 2, 3

For the East door with the same time period, the codes would be:

January	5, 2, 1, 3, 4
February	4, 2, 1, 3, 5
March	4, 5, 2, 1, 3
April	45, 2, 1, 3

For the West door with the same time period, the codes would be:

January	5, 3, 1, 2, 4
February	4, 3, 1, 2, 5
March	4, 5, 3, 1, 2
April	45, 3, 1, 2

Variations may be chosen, and a much simpler formula, or a more complex one may be used. However, the important point is that the door users can determine the combination by a simple formula, as in the above example, by knowing the month, the door position, and the seed from part 3). This seed changes once every four months. The rule of part 3) could instruct one to push all buttons not used so far from highest numbered to lowest numbered, or to push all unused buttons at the same time.

It may take a bit of time to become accustomed to this method of code creation, but if used over the year, it would be predictable to the users and only minor updates would need to be sent out every four months. If the coding scheme became known to outsiders and a security problem developed, it would be a minor adjustment to change the code (for example, reposition the month and door orientation order or some other such minor change). In addition, students needing to access the room may be given the combination only once a month, reserving the coding scheme for the instructors.

It is conjectured that the odds of guessing any formulated scheme if the formula is not known is the same as having a totally random code. As more of the formula is known, the odds of guessing it decreases, however, this is the trade off, in order to increase usability (getting into the room desired when one wishes to) of the system for the users.

3.2 Network Security - A More Complicated Combination Door Lock Applicable to Communications Network Security

Consider that passwords and usernames when interpreted by the computer end up being nothing more than numbers, and even encrypted passwords are just more numbers filtered through algorithms. The same principle of the door lock algorithms could, therefore, be applied to networks. Why would you want to? Consider the case where a distributed WAN has hundreds of servers and superuser level access is required by single server managers (for their site only) and a central team that assists in managing the entire WAN need access to all the servers. Also, assume that all the passwords (different for each server, to prevent local superuser managers access to each others' servers) need to be changed on a regular basis. This could be done predictably using an algorithm and a program that changes the passwords then e-mails the single new password to each local server manager. Since the central team knows the algorithm, they can determine the password to any server.

3.2.1 Getting More General - Time to Get Through the Door

It starts to become obvious at this point that a general (algorithm) theory could be extracted and then applied to a number of cases as needed. This especially makes sense when security is required in a predictable way in a large set of components. Two sets are defined, a local set and a global set. The members of the set must have the following characteristics (intension):

Local Set:

- each element allows unique or limited access locally;
- local access values must change over time;
- the extension of the set does not presuppose exact intension of the set, only similar intension.

Global Set:

- one element is allowed unlimited access;
- global access values must change;
- the extension of the elements (subset) does presuppose exact intension of the whole set;
- predictability is essential.

A different way of describing this relationship is to say that the global set is actually the algorithm that best describes all of its elements, rather than a subset of them, with global access. To model a set simply identify characteristics of each network element, find the relationships of the elements to construct the security algorithm and apply it by adding a predictable known variable for the seed.

3.2.2 Doing Things Backwards

If the relationships are known, but elements are missing, perhaps lost members could be reconstructed. This appears to be the method utilized in reconstructing failed drives in strip drive arrays with the help of XORed parity drives and, when data is reconstructed on disks with off the shelf data recovery programs. To recover lost data on a distributed network, the local relationships need to be known. For a single program like a knowbot to function in more than one site, it must contain global relationships or rules, common to the network's data (if applicable). If the DEFINE language could be used, it would need to possess the capability to define local and global relationships of the characteristics described by J. Firor.

Therefore, we have observed that set intension can determine the set extension predictability.

4 GST Rebate Sniffer Project

The objective of this research is to provide the ability to reclaim data that has been recently erased in a computer network. While it is impossible to unshred recently shredded paper documents, we will be able to restore to readability documents that have been deliberately hidden in the network on short notice. This research has general use in surveillance applications such as law enforcement and taxation. To illustrate, consider the Goods & Services Tax (GST). Revenue Canada attempts to ensure that revenue producing companies have rebated all the GST that they should. Auditors enter companies with short notice during which time the company may attempt to hide some of its information inside the computer network. The auditors are equipped with devices which probe into the network to view information about the company’s transactions. But what about those transactions that the company has just erased? This project involves the development of tools that would provide the auditor with the ability to view transactions that have been recently hidden in the network. In surveillance applications, there is a certain time period during which the hidden data can be recovered and after that they are permanently deleted. Thus it is important that the auditors get to the hidden data before it is too late. A network sniffer monitor permits data in the network to be viewed instantly. There are two reasons for erasing data: first, because some group has approached your network that you do not want to see all of your data and, second, by accident. The sniffer based tools also provide a safety measure, that is, a backup system for security in case of accidental deletion. In summary, this research involves writing application specific software for 1) surveillance and 2) security.

4.1 Research Program

A network sniffer monitor is a portable computer that looks like a suitcase with a cable that can be plugged into a computer network and a screen that permits operations in the network to be viewed. The sniffer includes software that permits summary tables and graphs to be printed and artificial intelligence that permits analyses of the data before printing. A network sniffer monitor is useful for the development of user written protocols for hiding data and recovering hidden data in a computer network.

The Network General network sniffer monitor (provided by Atalco) is crucial to this project. Both portable and distributed sniffers are needed. The **portable expert sniffer** is advantageous for the collection of short-term data because it can be easily moved to different sites. It is essential for studying hidden transactions in surveillance applications. However, the portable sniffer only "sees" data on the segment (or ring) that it is attached to. In other words, if we are attached to segment A in the computer services building, we cannot see data that is local to segment B in the economics building if there is a device (such as a bridge or router) that is blocking segment B data from getting on the network (and thus to the sniffer). The main advantage of the **distributed expert sniffer** (DSS) is that it allows users to perform monitoring over a distributed network from one site. The DSS permits long term data collection and analysis. It is expected to be busy for long periods of time while the portable sniffer is available for site visits. For the GST application, the DSS together with the portable sniffer permits construction of a realistic test environment: the distributed sniffer simulates the communications network of the revenue producing company who is attempting to hide some of its transactions while the portable sniffer can be easily moved throughout the network monitored by the DSS. It is important to capture the notion of competition in our test environment. Both the auditor and the company being audited have their network monitoring devices. The DSS is busy generating new transactions attempting to ensure that it does not overwrite hidden transactions until the auditors leave while the portable network monitor is interrogating the network.

The objective of Johnson's National Sciences and Engineering Research Council of Canada (NSERCC) funded research program (**Natural Language in a Network**) is to provide English language access to databases where the natural language interface and the database are distributed in a network. We map a user's conceptual view of a network into the processes involved in interpreting distributed natural language database requests. Graduate student Lin Chen, has provided database modelling of the user's conception of concurrent network processes. Her work (Chen & Johnson 1994a; 1994b; Johnson & Chen 1994) is relevant to this research project because it addresses the development of communications protocols at the user level in the OSI framework. User written protocols for hiding data and recovering hidden data in a computer network occur at this level.

5 Conclusion

In this paper, we have described research involving university, government and industry in the area of network communications. Our specification language for describing networks is DEFINE (Gilmore 1987; 1988) which was originally developed for specification of a database for enterprise modelling. We are interested in where the language DEFINE breaks down when put to the task of network modelling. We are interested in network modelling because, for large networks, management is difficult due to an inability to model them.

Combination door locks algorithms illustrate the notions of intension and extension of a set and introduce the possibility of a changing set intension. Derzaph says, "The odds of guessing any formulated scheme if the formula is not known is the same as having a totally random code (assuming no observations have occurred)." This can be restated by substituting the word "extension" for "scheme" and "intension" for "formula". Derzaph's quote is restated as follows: if we do not have the intension (combination, formula) of a set, we cannot determine its extension (formulated scheme). The lock combination changes with time. The intension (combination, formula) changes with time, but less frequently than the lock combination.

We have discussed our design of software for a "network monitor". The plan is to develop a software program for surveillance use by the auditors of Revenue

Canada. Arriving unexpectedly at a place of business which they suspect to have been evading full payment of their taxes and armed with a network monitor, also known as a "sniffer," and network sniffer software, they will be able to recover recently erased financial information which the proprietors of the business may have attempted to delete from their own networks as the auditors arrived at the door.

Firor concluded his article as follows: "Although this project started with grand schemes and ideas but ended in failure it was an excellent experience in research. I was given a chance to review what is taking place in the Telco industry with respect to networks. I was able to take aim at a little chunk of network management and apply a new concept to it. Even though it was deduced that DEFINE was insufficient, reasons were found for it being that way. Pursuing these reasons was challenging and rewarding in itself."

However, Firor models networks by insisting that their structure parallel the structure of the expression needed to describe them. He provides an approach within which it would be possible to construct a theory of network processes. He succeeded in identifying where the current theory of network processes (embodied in the language DEFINE) need refinement. Gilmore has reached a similar conclusion, that is, that an artificial restriction on abstraction imposed by DEFINE hindered it application.

Johnson’s work with graduate students (Chen supported by NSERCC and Abouzgaia supported by Telecommunication Research Labs) involves generalizing all of her previous applications of Gilmore’s set theories (Johnson & Rosenberg 1995; 1992a; 1992b; Johnson 1995; Johnson 1993) to operate in a telecommunications network. Relation nets are a class of Petri net which offer a formal treatment of individuals and relations in dynamic system modelling. In addition, system property verification can be achieved by simulating a relation net. In (Chen & Johnson 1994a; 1994b; Johnson & Chen 1994), the relation net representation is applied to to parallel processing of database queries. A Static Set-Relation net (SSRN) is proposed to represent the underlying set schema of a database in a net structure. A Dynamic Set-Relation net (DSRN) is then established to simulate a distributed database query.

6 Appendix

Computer and telecommunications networks are currently growing at a rapid rate in terms of both size and complexity. Traditionally these networks have been managed by humans who relied on their training, instinct and past experience to deal with the problems that have arisen. The evolution of these networks has now reached the point where management and administration of these networks has become a difficult if not impossible task, in some cases, for humans to perform. There are several reasons for this, including:

- the high rate at which network events occur;
- the complexity of the network situation;
- the need to take management/administration actions quickly.

It is evident that computers represent an ideal solution to these network management issues. Before systems can be designed to perform automated network management and administration several areas of research need to be addressed. Some of these requirements are:

1) to express in concise language the basic properties of computer and telecommunication networks;
2) to express these properties in a format understandable by computers;
3) to define methods and standards with which network statistical and diagnostic data may be reported to the system;
4) to give the system an expert knowledge base of computer network management and administration;
5) to apply AI techniques to solving the issues and circumstances with which the system will be faced;

Ideally any network management program should be able to run in a distributed mode. That is the management takes place concurrently at several different sites. The reason for this is simple: the time delay involved in communicating with remote sites may not be acceptable in many situations.

It should be noted that the SNMP and CMIP standards have defined the format and the method of reporting network diagnostics and statistics. There are currently several network management products in use by industry. They suffer from several drawbacks:

- network managers require vast knowledge of many different networking aspects;
- these packages do not run in an unattended distributed mode;
- the human delay in recognizing the problem, formulating a solution, and taking corrective action is unacceptable.

Our research is to address requirements 1 and 2 as listed above. We also discuss several ideas regarding the actual administration of computer/telecommunications networks. Most of the areas addressed in this research are in the lower levels of the OSI model.

6.1 Preliminary Ideas

1. Signal types

There are two basic (primitive) signal types, analog and digital. More complex types may be created from these two basic signal types such as multichannel digital signals.

2. Channel types

Channels are the specific paths along which data signals flow between two elements. Channels may carry more than one signal (ie. multiplexing).

3. Multiplexing

There are two types of multiplexing; time division where each signal occupies a certain time interval, and frequency division where each signal occupies a certain portion of the frequency spectrum.

4. Elements

Elements are entities from which data is transmitted, received or both. Networks themselves may also be elements. The advantage of allowing networks themselves to be elements is that one may look at networks at multiple levels of abstraction. Some examples of network elements are network cards, networked computers, routers, bridges, multiplexers, switches, hubs, etc. Elements may be part of more than one network. Elements are connected through links.

5. Transmission media types

Some of the transmission media types to be considered are copper wire, twisted pair wire, coaxial cable, fibre optic cable, and microwave links (terrestrial and satellite). The inclusion of these types is important so as to give the model a clear understanding of the properties and limitations of these media types. If these are not included, situations could arise where a computer may not realize that the physical constraints of the media have been exceeded. A computer model must have some representation of the inherent physical properties of the network.

6. Network topologies

The network topologies of interest are point-to-point, star, bus, hierarchical, ring, unidirectional ring, bidirectional ring and meshes. The reader is referred to any introductory communications network textbook for a description of the topologies.

7. General

Upon completion of the topology modelling, we expect to pursue modelling of network elements such as routers, bridges, switches, etc. Additionally, it appears worthwhile to pursue modelling the media physical characteristics. Although this work is very basic, we believe that if we model the basics correctly, the complete model will be clear, concise and unambiguous. Using artificial intelligence we will be able to draw inferences from the model and make decisions based on these inferences. These decisions could include rerouting network flow, disconnecting network components, minimizing cost, preventative maintenance, reconfiguring elements, identifying critical paths and components, and generating reports for humans.

Some of the background work for this project is already in place. The ANSI standard T1.214 outlines the interface between network elements and operation systems. CCITT recommendation X.721 outlines the structure of management information. There is also the recommendation of technical subcommittee T1M1 which outlines a methodology for developing services and protocols for Telecommunications Management Network (TMN) applications. These technical recommendations and standards outline the structure of reported data and services, the Protocols used in communication between OSes (operating systems) and NEs (network elements), and the functions of a network’s management components. This proposed research is not so much to provide alternatives to these ideas but rather to enhance their use. To the computer, all the data reported to the OS is meaningless. The computer requires some kind of model and some set of "rules" on which it may apply this data. The idea proposed is shown in Figure 6, an Intelligent Management Engine (IME) which acts on the reported data, and uses it's expert knowledge and model to make decisions. Although the approaches used in the computer and telecommunications fields are different, the basic model developed here may be used in both.

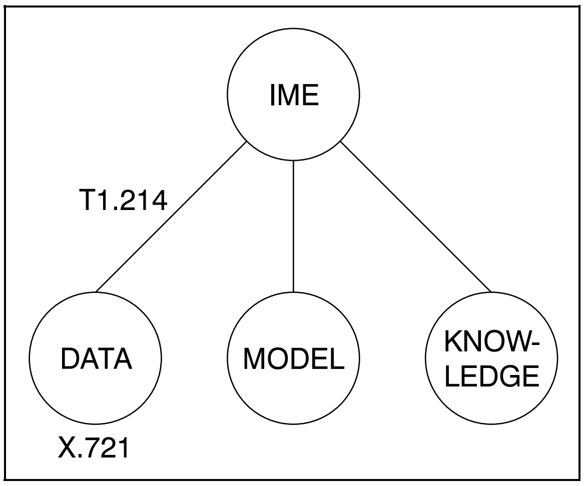

Figure 6

There is an alternative to the object oriented approach proposed by the telecommunications industry. In some time critical applications involving geographically remote sites, rapid action is called for on the part of the network operations and administration staff and/or OS. The data and actions required may not be able to be transferred within an acceptable time period. The alternative would be to send a "knowbot" to the network element(s) in question. These knowbots would be self contained entities which can be transferred to a site, make a diagnosis and take appropriate action. Upon completion of the task they return to their host and report their findings and actions. A parallel to this approach in the real world is that of sending a repair crew to fix a damaged item. The repair crew has the ability to gather information, draw inferences based on their knowledge, take action and report back to their company when the task has been completed. Knowbots may also remain resident at certain "depots" on the network from which they may be dispatched. The advantage of having knowbots at depots is that the entire knowbot does not have to be transferred to a geographically remote site, only the signal to dispatch it has to be sent. The knowbot simply travels the short distance from depot to site (Figure 7). The knowbot approach may be used in situations where it is not feasible to have management programs present at all times, otherwise the distributed approach may be the best plan.

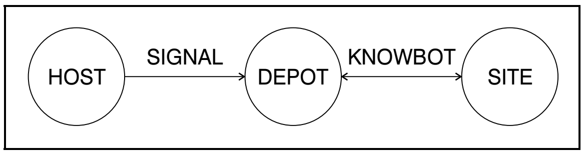

Figure 7

References

1. Abouzgaia, S. M. (1994). An approach to knowledge-based integrated network management system. Unpublished manuscript, University of Regina.

2. Chen, L. & J. Johnson. (1994a). Petri net simulation of flexible manufacturing systems. In G. Rzevski, R. A. Adey & D. W. Russell (Eds.), *Applications of Artificial Intelligence in Engineering IX*, pp. 49-56. Southampton, UK: Computational Mechanics.

3. Chen, L. & J. Johnson. (1994b). Relation nets and database query. In *Proc. of the 3rd Pacific Rim International Conference on Artificial Intelligence PRICAI'94,* pp. 752-758. Beijing, China.

4. Gilmore, P. C. & G. K. Tsiknis. (1993). Logical foundations for programming semantics. *Theoretical Computer Science III*, pp. 253-290.

5. Gilmore, P. C. (1987). Concepts and methods for database design. Technical Report TR87-31, Department of Computer Science, University of British Columbia.

6. Gilmore, P. C. (1988). A foundation for the entity relationship approach: How and why? In *Proceedings of the 7th International Conference on Entity-Relationship Approach*, pp. 95-113. North-Holland, Amsterdam.

7. Johnson, J. & L. Chen. (1994). Relation nets for production scheduling in manufacturing. In G. Rzevski, R. A. Adey & D. W. Russell (Eds.), *Applications of Artificial Intelligence in Engineering IX*, pp. 585-592. Southampton, UK: Computational Mechanics.

8. Johnson, J. (1995). Semantic relatedness. *Computer & Mathematics with Applications*, 29(5), pp. 51-63. Pergaman Press.

9. Johnson, J. & R. S. Rosenberg (1995). A data management strategy for transportable natural language interfaces. *International Journal of Intelligent Systems*, 10(9), pp. 771-808. John Wiley & Sons.

10. Johnson, J. A. (1993). Optymalizacja bez Obliczen: Zastosowanie Sztucznej inteligencji do planowania organizacji produkcji (Optimization without computation: An application of artificial intelligence to product scheduling in manufacturing). *IV Krajowa Konferencja Robotyki (Fourth National Conference on Robotics Vol. 1),* pp. 235-242. Wroclaw, Poland: Instytut Cybernetyki Technicznej (Institute of Engineering Cybernetics), Politechniki Wroclawskiej (Technical University of Wroclaw).

11. Johnson, J. A. & R. S. Rosenberg. (1992a). Knowledge sharing for natural language understanding and conceptual modelling for database design. In *Proc. of the 2nd Pacific Rim International Conference on Artificial Intelligence PRICAI'92*, pp. 980-988. Seoul, Korea: Center for Artificial Intelligence Research.

12. Johnson, J. A. & R. S. Rosenberg. (1992b). A measure of semantic relatedness for resolving ambiguities in natural language database requests. *Data & Knowledge Engineering,* 7(3): 201-225.