Towards a Self-Healing Network in Controlling Access to Network Applications

Abdullah Gani and G. Manson
University of Sheffield, Sheffield, UK

agani@ieee.org gam@dcs.shef.ac.uk

Abstract

A self-healing network is a 'dream' for every network designer today. Abstractly it is a network that capable of maintaining the availability of network services. This would not be impossible to be turned into a reality if the networks are embedded with intelligence. By having the intelligent capability the network can be self-managed in response to events that taken place within the network itself through the learning mechanisms. This paper presents the User Manager Agent System (UMAS) learning performance in controlling a request for network application session. ANFIS of Matlab v5.3 has been used to simulate the learning process in which it bases on the training data of network state. The work also had revealed some weaknesses of ANFIS in processing a large volume of training data sets and substantial amount of time taken for processing.

Keywords: Intelligent Agent, Neuro Fuzzy Logic, and Network Application Management

Introduction

Fundamentally, a network is established for providing services and applications to the users, therefore, the network should be user-oriented to fully cater the needs of the users. However, the demand for having a wide range of applications has never had decreased instead it grows exponentially due to a number of reasons. Networks, on the other hands, are expected to be capable of fulfilling all the demands and maintain the level of services that had been provided before. While maintaining the service availability, the network is also expected to be able to cope up with changes in the user's requirements. This paradox has led to the requirement of having the control mechanisms for achieving those goals (Peterson, 2000).

A network is incomplete if no services are offered. However, in providing those services, resources are needed. Users expect the network services should be available and run at an acceptable level of Quality of Service (QoS) (Wang, 2001). On the other hand, the administrators have a responsibility of controlling the usage of network resources toward productive and legitimate usages. These goals never had accomplished satisfactorily for both the users and the administrators. One of the reasons is that the network resources are always in short supply and costly to acquire and to maintain. Competing for resources by applications in the network can cause resource exhaustion. The term of resource exhaustion here refers to a state of which the network cannot provide more services in order to run an application. This is because the available resources have reached the threshold values that disallow more requests to be granted.

This paper presents an intelligent agent system that capable of performing control mechanisms on a user's request for a network application. Intelligence here refers to an ability to produce a desirable behavioural outcome of maintaining network resource state equilibrium. In other words, it is an

Paper Accepted as a Short Paper

ability to process the request accurately and dynamically.

In the next section, an overview of the problems that related to the network application are presented and followed by the description of UMAS.

Network Applications

Generically, network applications are the software that runs in a network environment to get jobs done. For example, an application for sending messages through the network infrastructure is a network application. Network applications can be perceived by different schools of thought. One might perceive that network applications are applications that only can run if the network infrastructures exist. Others might perceive differently and to them network applications are applications that the network can offer including local applications such as tools for a host. Despite differences in way of defining network applications, both shares the same common feature that is network application require network functions to run. Network function is system software that enables the applications to be run in the network.

Nevertheless, the discussion of network application is incomplete if the network resources are not included (Croll, 2000) . This is because the network applications are related to network resource utilisation, which is a requirement for a network application to be functioning. In other words, a request for a specific network application is effectively meant requesting network resources, as they are interdependence. In an email system, when a user wishes to send a message to the addressee, the user has to make a request for a number of network resources for processing and transmitting the message. Similarly to a request of accessing a web site, a number of resources are needed such as bandwidth as a 'transporter', services in term of protocols, processors in term of hardware and software.

In addition, the availability of network applications is determined by the service existence. This is because services are a prerequisite requirement for a network application is to be available. Technically, network applications are processes that communicating and a process can be perceived as a program that is running within end system (Kurose, 2000). For example, when a user wishes to send a message electronically, certainly services are needed to forward packets of message to a Mail Transfer Agent in a server from a User Agent that is located in a workstation (Raghavan, 1998).This service is actually attained from a number of processes that are communicating with each other to enable the message packets to be forwarded. Plainly this shows that network applications depend on services in order to be available and functioning. In short, we can say that network resources determine the network services and network services in turn determine the network applications or via versa. Let say, a collection of resources, R, is represented by $\{r : r \in R\}$; a collection of network services, S, is represented by $\{s : s \in S\}$; and a collection of network applications, A, is represented by $\{a : a \in A\}$. Despite that $A \neq S \neq R$ but $R \rightarrow S, S \rightarrow A$ and this would be become a premise for the research.

Interdependence between network applications and network services however has created the problem of performance deterioration in which network applications have a slow response to a request due to the required resource has reached a threshold level. This can be avoided if intelligent agents are deployed to learn about the condition of the network and to make decision according to the current situation of the network. In the next section, a proposed system is introduced.

UMAS Description

Generically, UMAS is an interface between users and networks for the purpose of controlling network resources as a result of user's request of network application. Figure 1 shows a high level diagram of UMAS with the example of network resources. The Router and server are the examples of network resources and both are required by the network applications.

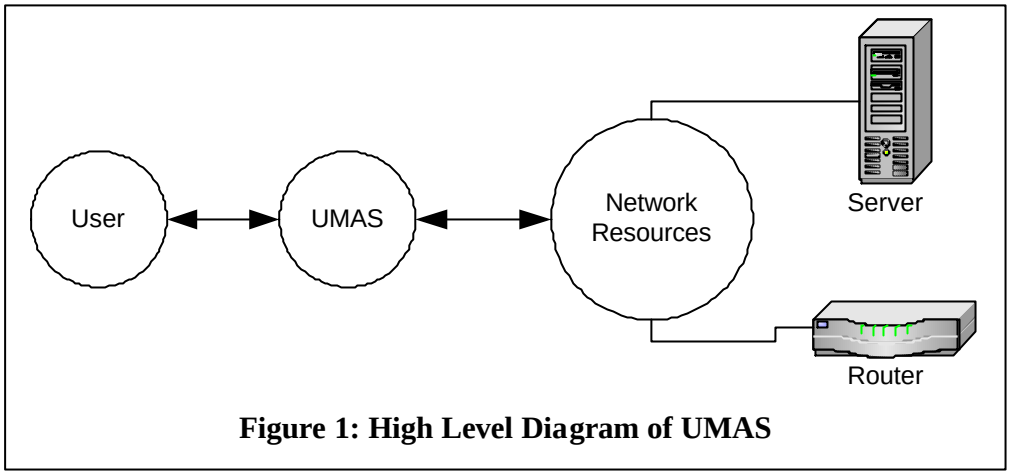

Figure 1: High Level Diagram of UMAS

The UMAS determines the access time of a session for a particular user so that the network resources of bandwidth can be allocated accordingly to meet the requirements of application and Quality of Service (QoS). For example, if a number of users accessing network resources are relatively small compared to available resources, then even the least critical applications can be allowed to access the network resources. However, if the number of users increased, then the least critical application sessions will be discarded from the network.

When a user logs into a network, the UMAS will read the *profile* file of the user and retrieve the current available knowledge of the network form a knowledge base. A Knowledge base is a repository for the learned knowledge of network element usage such as router, bandwidth, servers and applications. A Profile file is created by a system administrator and it contains a collection of user settings such as desktop attributes and allowed network connections (Sosinsky, 2000). This knowledge is critical in the process of granting access to the network application request because it helps the UMAS to decide the best option for a network to be utilised for a particular session. The decision making process for determining a session period will be updated periodically at the interval time of 100 ms., which is believed to be sufficient for establishing communication between UMAS with other network objects.

The UMAS consists of several agents which each of them are assigned with a relevant task. Figure 2 illustrates the main components of the UMAS in which are actually intelligent agents. The four components are Knowledge Learner, Granter, Enforcer and Verifier. All the components are governed by the manager that resides in the UMAS.

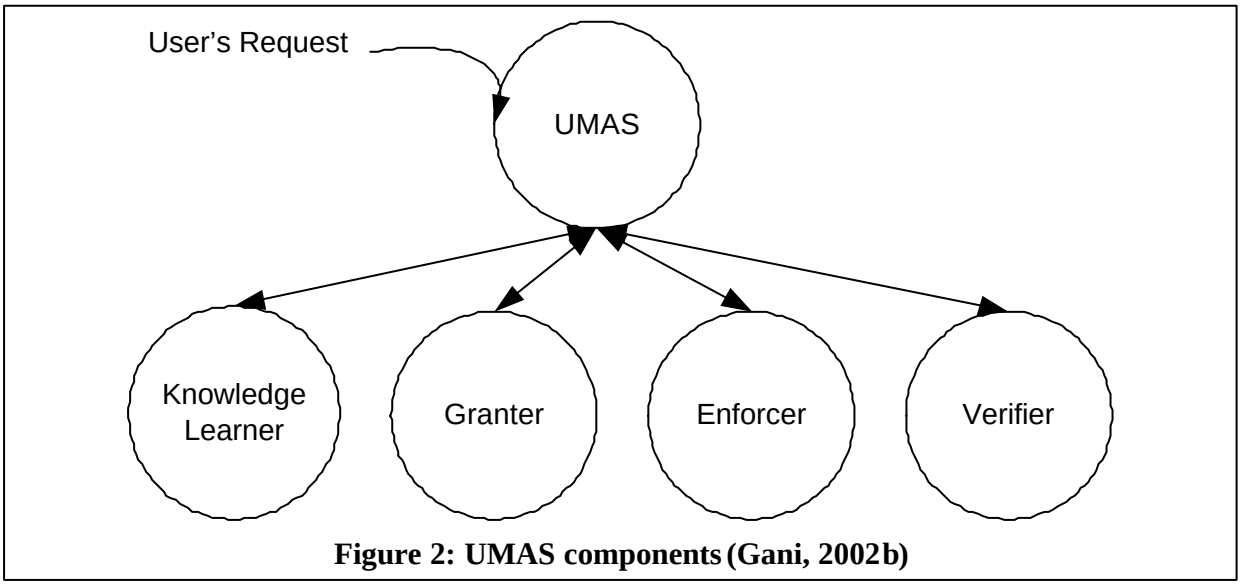

Figure 2: UMAS components (Gani, 2002b)

The Knowledge Learner is the core component of UMAS and is responsible for capturing a set of variables so that they can be stored in the repository for future retrieval. When a request is received by the UMAS for accessing a particular application, it needs to be processed with the consideration of network resource availability and its criticality to the organisation's objectives. The UMAS forwards the message of request to the Granter for a decision whether to approve or otherwise. In deciding on the request, the Granter relies on information about the users that are currently logging in to the network and the network resource usage. These pieces of information are necessary in order to determine the consequences of potential approval upon the network performance. The Granter will use Fuzzy set data supplied by the Knowledge Learner to compute the time that practically can be allocated for the session. The decision taken by the Granter then will be forwarded to the UMAS for execution.

The component of Enforcer is responsible for executing the decision by communicating with the routers to inform them of changing packet priority. At the same time, the Granter will set the clock on and start counting. However, the Granter needs to inform the Verifier as well because it is responsible for verifying the decision by checking the user profile with the server. If the allocated time is 'correct' then the user can have such application to be accessed otherwise a new computation of time will be asked. Of course, the UMAS needs to be informed about any decision taken by both the Granter and the Verifier, as it needs to inform the user too.

User Manager Agent System (UMAS) also acts as an intermediate filter mechanism between network and the users for achieving equilibrium state of network in providing the best level of services in term of network service responsiveness. The UMAS contains several independent intelligent agents that each has a specific task in which under the control of agent manager. Figure 3 shows a high level diagram of UMAS in which it has an ultimate goal of coordinating goals of user and network fulfilment. Obviously, users use the network applications without knowing the state of the network. In many cases, certain servers are heavily used whereas other might be idle. Similarly certain application may heavily requested and others may not. So, UMAS will facilitate both ends goals with three sub-goals – execute management actions, process access request and build knowledge base. Hexagon shape represents task that needs to be performed in order to achieve the goal.

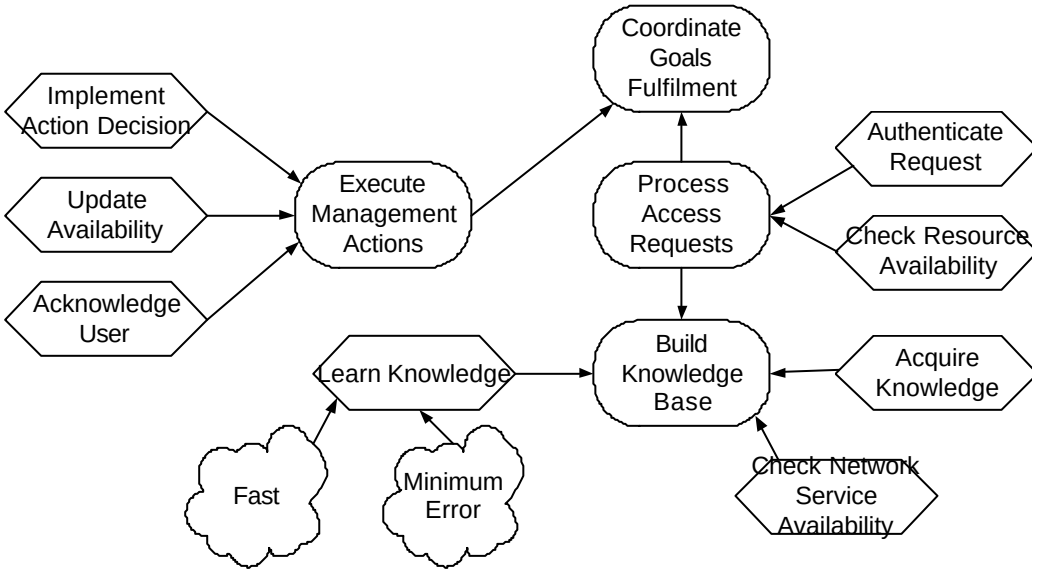

Figure 3: Goal Diagram of UMAS (Gani, 2002*a*)

The UMAS will receive a request for a network application and processes it accordingly. The request has to be authenticated in order to protect legitimate request. Other agent checks the resource availability by interfacing with the Learner agent. Both agents are executed synchronously. An action decision is passed over so that it can trigger the learning about the network to take place.

UMAS Learning

As depicted in Figure 3, the UMAS needs to learn knowledge from the Knowledge Base (KB), which is a repository of training data sets. The task of Learn Knowledge has two soft-goals - fast and minimum errors, which are the target of the UMAS learning capability. These features are crucial in determining timely decision that can be taken. As the size of training data sets are relatively large and contain a number of sets, the learning processes need to be executed in stages. Besides, the output of a particular learning process is an input to another learning process. A unit of processor handles only a particular set of training data in order to make the learning is efficient.

The UMAS processes the request for granting a permission of accessing a particular network application based on several parameters such as user's role, application criticality and bandwidth condition. Every session is granted in term of period duration, which is depends on the network load. For example, if many users heavily use certain application, plainly an additional request will make the level of service deteriorates. Similarly the throughput rate will drop substantially to a level of service that cannot be accepted by the users. Therefore, it is necessary to have an intelligent learning about the network load before permission of the session can be granted.

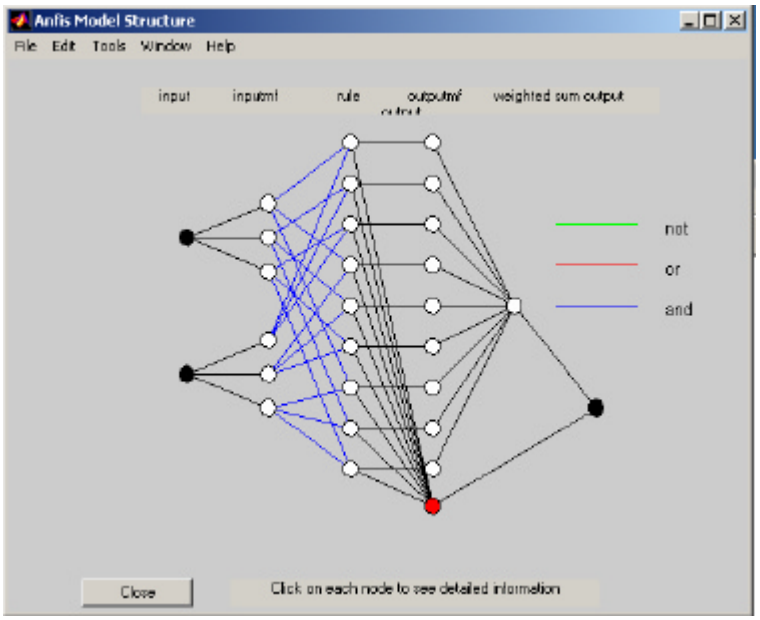

Figure 4: Neuro Diagram of UMAS

For the learning purposes, the UMAS uses a Neuro Fuzzy Logic (NFL) algorithm in which it contains a combination of fuzzy logic and neuro network algorithms. This combination has enabled the rule sets are extracted from the training data automatically by ANFIS of the Matlab software. The UMAS has to establish a reliable learning mechanism before it can be integrated into the development (Hopgood, 2001). The training data sets consist a numeric representation of number of users, throughput and application criticality.

Figure 4 shows two inputs that have been used and each has three membership frequencies - low, medium and high respectively.

Simulation

The UMAS is a system that requires a great number of interactions with many agents to be fully functioning. However, the main factor in which it relies will be on the algorithm used to do the processing especially in decision making for allowing a user(s) to have certain period of accessing the application. Fuzzy Logic is used to help the Granter to make a decision upon the request received from a user(s). As the Granter needs to compute the time for allowing a user to access a particular application, it requires a

set of Fuzzy set data. Firstly, the Granter needs the value of the applications' criticality. The criticality of the application can be represented in a range of values (1-10) in which the higher value denotes higher criticality. However, the demarcation of criticality is quite fuzzy because certain application can be quite critical during weekdays and become casual during weekends. Secondly, bandwidth availability is another variable that needed in the computation of time. The 'Fuzziness' of the bandwidth availability can be viewed in terms of what is the best throughput rate for application for smooth execution. Again, this depends on the application requirements.

	Low	Medium	High
Bandwidth Availability (BW)	0 – 35	36 - 60	60 - 100
Application Criticality	0 – 3.5	3.6 – 7.4	7.5 - 10
Allocated Time (h)	0.5	1.00 – 2.00	2.00 – 4.00

Table 1: MF values assignment

Both variables are divided into three membership frequency (MF) – low, medium and high. Table 1 shows the assignment of values for all MFs. All the values were assigned according to the current policy requirements and they are subject to changes in the requirement.

MATLAB® has been used to simulate a few scenarios to automatically allocate users with different times depending on the application criticality and BW availability. Figure 4 illustrates the first extreme scenario in which both input variables are in a low state and it shows that the access is relatively a short period. This indicates that both variables have a significant correlation towards the time allocated.

The simulation was conducted by Matlab ver 5.3. and the purpose of simulation was to establish rules with minimum error rate that will be integrated into the UMAS learning mechanisms. In achieving this, the training data that was collected using Windump and Windows Network Monitor tools, need to be converted into numeric format before being put into use. This is because of the ANFIS, the learning algorithm of neuro fuzzy logic can only accept numeric representations and limited input (Cox, 1999), (Knapik, 1998). For that reason, the simulation was based on two set of inputs – application criticality and user priority. Both inputs were given a similar scale of 1 – 10. A smaller number for application criticality denotes the higher criticality it was.

Figure 5 illustrates the rules that have been automatically created by the ANFIS, which shows the allocated time is 100% if the application criticality and user priority are high. On the other hand, the allocated time drops substantially if both inputs decrease. However, because of the criticality was given higher weighting, any changes in the values significantly affect the outcome of the allocated time.

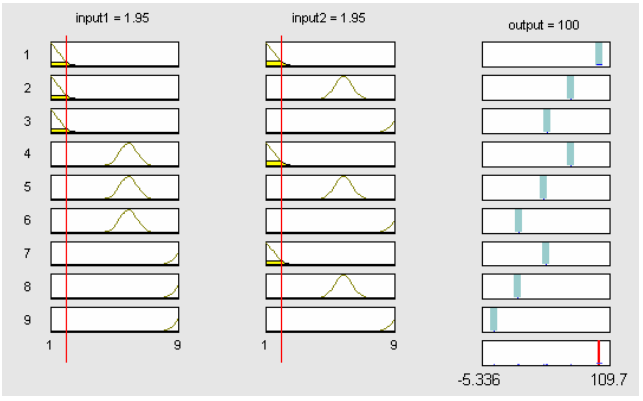

Figure 5: Generated rules

Figure 6 is the results of the simulation in which it clearly shows that all data were distributed into three clusters – high, medium and low. Most the data were scattered in the medium interval and with minimum error rate. Figure 7 shows the results of simulation that indicate learning had taken place with the error rate that can be accepted

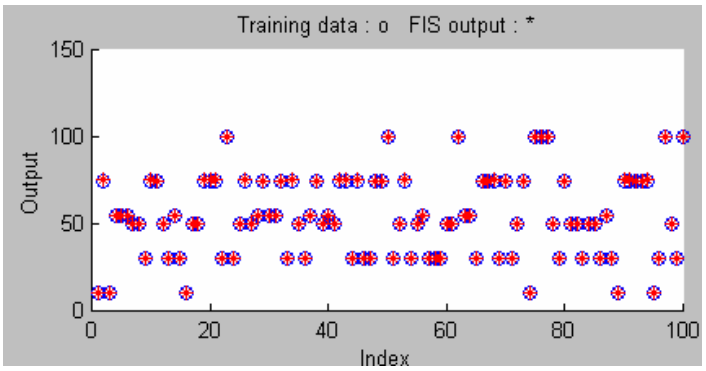

Figure 6: Results of the Simulation

Epocs	Error rate
3	0.0000393
100	0.0000618
200	0.0000714
500	0.0000618

Figure 7: Error Rate Results

Conclusion

The undertaken simulation had generated a set of rules with minimum error rate that can be integrated into the UMAS agents' development. The outcome of the simulation can be used for the UMAS agent to decide the best allocation of session time for a user.

The UMAS has the capability of making intelligent decisions based on the condition of the network and on the user's roles by performing learning. This enables the network resources to be allocated for the achievement of network establishment objectives and the organisation's goals. At the same time, users can receive an appropriate level of network services, in term of application availability and satisfactory network responsiveness. As the UMAS has the learning capability, the Knowledge Base will be gradually getting larger and this could lead to higher intelligence attainment of the UMAS. The idea of UMAS also can be applied into other network management activities such as traffic management, Quality of Service and as a 'teacher' for training an agent that is newly introduced to the network.

Reference

Cox, E. (1999). *The Fuzzy Systems Handbook - A Practitioner's Guide to Building, Using and Manipulating Fuzzy Systems.* AP Professional.

Croll, Alistair and Packman, Eric (2000). *Managing Bandwidth: Deploying QoS in Enterprise Networks*. New Jersey, Prentice Hall.

Gani, Abdullah. et.al. (2002a). Developing an Intelligent User Manager System for Controlling Smar School Networks. *Malaysian Journal of Computer Science 15*(2).

Gani, Abdullah. et.al. (2002b). The Roles of Intelligent User Manager Agent for Controlling Access to Network Resources. 3rd. *Annual PostGraduate Symposium The Convergence of Telecommunications, Networking and Broadcasting.* John Moore Univ, Liverpool, UK.

Hopgood, A. (2001). *Intelligent Systems for Engineers and Scientists*. CRC Press LLC.

Knapik, Michael & Johnson, Jay (1998). *Developing Intelligent Agents for Distributed Systems, Exploring Architecture, Technologies and Applications.* McGraw Hill.

Kurose, James F.& Ross, Keith W (2000). Computer Networking - A Top-Down Approach Featuring the Internet. London, Addision Wesley.

Peterson, Larry & Davie, Bruce S (2000). Computer Networks - A System Approach. San Francisco, Morgan Kaufman.

Raghavan, S.V. & Tripathi, Satish K. (1998). *Networked Multimedia Systems - Concept, Architecture, and Design.* N.Jersey, Prentice Hall.

Sosinsky, Barrie & Moskowitz, Jeremy (2000). *Microsoft Windows 2000 Server in 24 Hours*. Indianapolis, SAMS.

Wang, Z. (2001). *Internet Quos: Architectures and mechanisms for quality of service.* San Francisco, CA, Morgan Kaufmann.