Decentralised Approaches for Network Management

Mohsen Kahani, H.W. Peter Beadle
The Institute for Telecommunication Research (TITR)
Elec. and Comp. Eng. Dept., University of Wollongong
Northfield Ave, Wollongong, NSW 2522, Australia
email: {moka|beadle}@elec.uow.edu.au

Abstract

Centralised network management has shown inadequacy for efficient management of large heterogenous networks. As a result, several distributed approaches have been adapted to overcome the problem. This paper is a review of decentralised network management techniques and technologies. We explain distributed architectures for network management, and discuss some of the most important implemented distributed network management systems. A comparison is made between these approaches to show the pitfalls and merits of each.

1. Introduction

Systems are increasingly becoming complex and distributed. As a result, they are exposed to problems such as failure, performance inefficiency, resource allocation, security issues, etc. So, an efficient integrated network management system is required to monitor, interpret, and control the behaviour of their hardware and software resources [1]. This task is currently being carried out by centralised network management systems, in which a single management system monitors the whole network.

Most of existing management systems are *platform-centred*. That is, the applications are separated from the data they require and from the devices they need to control. Although some experts believe that most network management problem can be solved with a centralised Simple Network Management Protocol (SNMP) [2], there are real network management problems that cannot be adequately addressed by this centralised approach [3]. Also, as managed devices are increasingly equipped with powerful processing resources, *device-centred* management can easily make use of these resources and provide a simpler, more scalable, more flexible, and cheaper management paradigm than platform-centred ones . There are a number of metrics that can be used to determine if a network management application is more appropriate for a centralised or distributed system. Figure 1. shows a metric presented in [3].

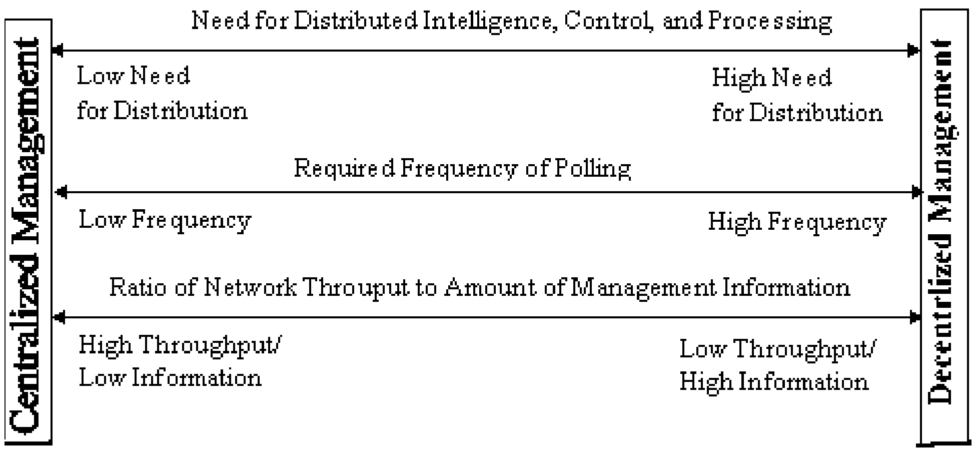
Need for Distributed Intelligence, Control, and Processing
Low Need for Distribution
High Need for Distribution
Required Frequency of Polling
Low Frequency
High Frequency
Ratio of Network Throuput to Amount of Management Information
High Throughput/ Low Information
Low Throughput/ High Information
Centralized Management
Decentrlized Management

Figure 1. Metrics to determine decentralisation

In this paper, we focus on non-centralised approaches for network management, and discus several architectures proposed to accomplish this purpose. We initially, explain different architectures for network management systems. Then, we present background information on distributed systems and then discuss several implemented distributed network management systems. Finally, we will conclude the paper with a comparison between these systems and discuss their pitfalls and merits.

2. Network Management Architectures

There are three basic approaches for network management systems: centralised, hierarchical and distributed [4]. Currently, most network management systems are centralised. That is, there is a single management machine which collects the information and controls the entire network (Figure 2.-a). This workstation is a single point of failure, and if it fails, the entire network could collapse. In case the management host does not fail, but a fault partitions the network, the other part of the network is left without any management functionality. Also, a centralised system cannot easily be scaled up when the size or complexity of the network increase [5].

A variation of centralised systems is the platform approach [6] (Figure 2.-b), in which a single manager is divided into two parts: the management platform and the management application. The management platform is mainly concerned with information gathering and simple calculations, while management application uses the services offered by the management platform to handle decision support and higher functions [7]. The advantage of this approach is that applications do not need to worry about protocol complexity and heterogeneity. The drawback is that it still inherits limited scalibility from its centralised architecture.
The hierarchical architecture uses the concept of “Manager of Managers” (MOM) and manager per domain paradigm[4, 6] (Figure 2.-c). Each domain manager is only responsible for the management of its domain, and is unaware of other domains. The manager of managers sits at a higher level and request information from domain managers. In this architecture, there is no direct

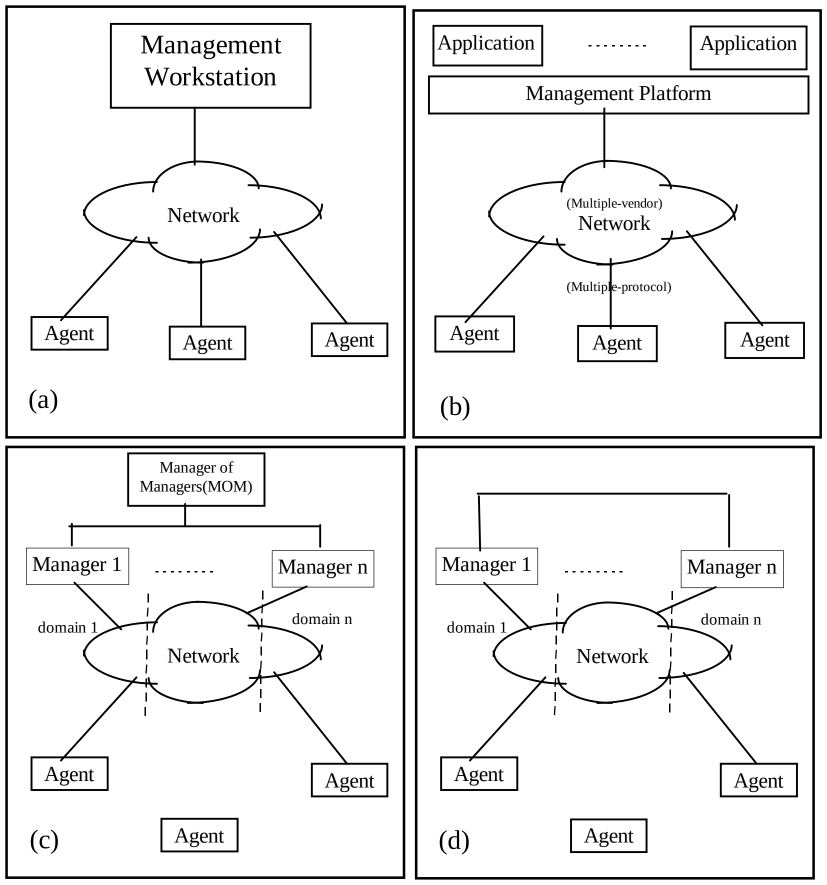

Figure 2. Different management approaches: a) Centralised, b) Centralised Platform-based, c) Hierarchical and d) Distributed.

communication between domain managers. This architecture is quite scalable, and by adding another level of MOM a multiple level hierarchy can be achieved.

The distributed approach (Figure 2.-d) is a peer-to-peer architecture[4]. Multiple managers, each responsible for a domain, communicate with each other in a peer-system. Whenever information from another domain is required, the corresponding manager is contacted and the information is retrieved. By distributing management over several workstations throughout the network, the network management reliability, robustness and performance increases, while the network management costs in communication and computation decrease [5]. This approach has also been adapted by ISO standards and the Telecommunication Management Network (TMN) architecture [8]. The Management model for ATM networks, adapted by ATM Forum, is based on this approach, too [9].

A combination of the hierarchical and distributed architectures, known as ‘network architecture’, can also be used [6] (Figure 3.). This architecture uses both manager-per-domain and manager of managers concepts, but instead of a purely peer-system or hierarchical structure, the managers are organised in a network scheme. This approach preserves the scalibility of both systems and provides better functionality in diverse environments.

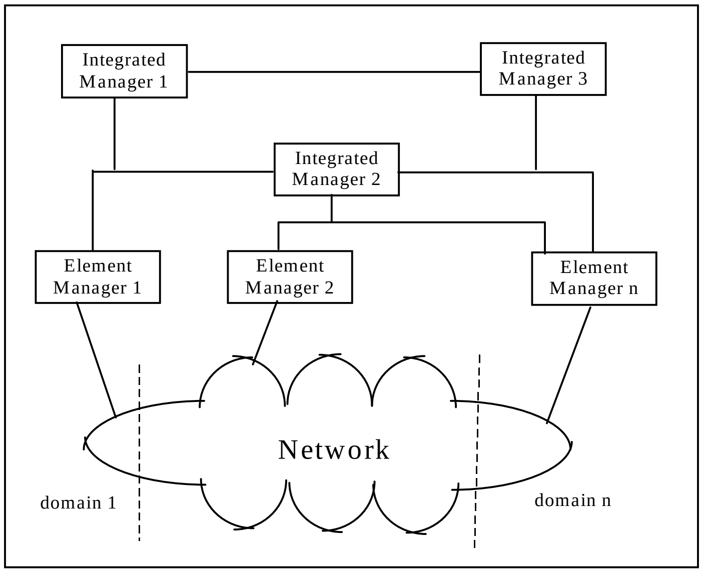

Figure 3. Network Architecture

3. Distributed Systems and Services

Fault tolerance and parallelism are key properties of a distributed system [10]. A distributed system should use interconnected and independent processing elements to avoid having a single point of failure. There are also several other reasons why a distributed system should be used. Firstly, higher performance/cost ratio can be achieved with distributed systems. Also, they achieve better modularity, greater expansibility and scalibility, and higher availability and reliability [11].

Distribution of services should be transparent to users, so that they cannot distinguish between a local or remote service. This requires the system be consistent, secure, fault tolerant and have a bounded response time. The form of communication in such systems is referred to as client/server communication [12]. The client/server model is a request-reply communication that can be: synchronous and blocking, in which the client waits until it receives the reply; or asynchronous and non-blocking, in which the client can manage to receive the reply later.

Remote Procedure Call (RPC) [13, 14] is well-understood control mechanism used for calling a remote procedure in a client/server environment. The idea is to extend the use of procedure call in local environment to distributed systems. This results in simple, familiar and general methods that can be implemented efficiently.

The Object Management Group's (OMG) Common Object Request Broker Architecture (CORBA) [15] is also an important standard for distributed object-oriented systems. It is aimed at the management of objects in distributed heterogenous systems. CORBA addresses two challenges in developing distributed systems: making the design of a distributed system not more difficult than a centralised one; and providing an infrastructure to integrate application components into a distributed system [16].

4. Non-centralised Network Management

In this section, we will discuss some of the most important proposed systems for management of networks using non-centralised approaches. These systems include distributed big brother, from the University of Michigan; Distributed Management Environment, from the Open Software Foundation (OSF); hierarchical network management, from the Technical University of Vienna (TU-Wien); and management by delegation, from the University of Columbia. A comparison will be presented in the next section. There are several other approaches, which are not discussed in this paper. Interested readers can refer to [17-24].

4.1. Distributed Big Brother

Distributed Big Brother (DBB) is a distributed network management system consisting of cooperative autonomous computing agents [5]. DBB has been designed with the goal of distributing management and organisational tasks, while maintaining a central location control over the whole system. It uses some technologies from the field of distributed AI, such as contracting information, organisational structuring, election for role management, and hierarchical control.

DBB distributes the management tasks by using mid-level managers that can be executed in parallel. It also uses the Contract Net Protocol [25] to reduce the overhead in authority structure. As the computer networks changes dynamically, it is not desirable to distribute the tasks only at time of initialisation. So, DBB allows the system to undergo an organisational self-design [26], and changes the submanagers roles as the network structure varies.

In DBB each agent consists of three basic functional modules: the communication process, the contracting process, and the task process. The communication process continuously checks whether a message has been sent to the agent, and if so, transmit it to the contracting process, which parses them and does appropriate actions. Finally, the task process carries out the agent’s management task.

Agents' roles are not static. An agent does not have any management role as it enters the network. At the time of initialisation, or whenever required, the agents choose a *chairman* to elect the *LAN manager*. The chairman announces the election and selects the best nominate (eg. least busy host) and broadcasts its decision. After electing the LAN manager, all other agents become *group managers* and execute tasks for information gathering. The LAN manager, which is responsible for managing the whole LAN, uses contracting to assign tasks to group managers. Finally, there is a fixed *top manager*, which uses contracting to request reports from LAN managers.

A network can have an arbitrary number of LAN managers and group managers. As LAN managers have some sort of autonomy and independency, more LAN manager means more autonomous and independent network management systems, which corresponds to higher robustness. Increasing the number of group managers increases the number of polling agents, which corresponds to a more frequent polling and more updated management information. However, there is trade off between increasing number of LAN and group managers and extra traffic caused by communication between top manager and LAN managers, and between each LAN manager and its group managers.

4.2. Distributed Management Environment (DME)

The Open Software Foundation's (OSF) Distributed Management Environment (DME) is an attempt to represent a structure under which the management of systems and networks can be brought together [27]. It tries to provide a standard set of services to bring management to everything, from bridges and router to server-based operating systems and applications [28]. DME is operating system-independent and supports several standards, such as SNMP and CMIP.

DME is tailored to OSF Distributed Computing Environment (DCE) [29]. DCE is a comprehensive and integrated service that allows development, distribution and maintenance of distributed applications. It masks the physical complexity of the networked environment and provides a layer of logical simplicity. DCE uses RPC for communication and provides distributed directory services, security services, time services and thread services. Using services provided by DCE, DME provides manageability and allows network components and applications to be well maintained.

DME consists of a framework that provides the building blocks for application development and some of the most critical system management functions, such as software management, licence management and printing services. The architecture of the DME allows the addition of new services. It consists of two key components: Manager Request Broker (MRB) and Object Server, as shown in Figure 4.

The MRBs are central pieces in the DME framework; they implement the basic APIs that access the services furnished by the DME. The MRB establishes a standardised programming interface to SNMP and CMIP, known as Consolidated Management API (CM-API). The MRB also provides

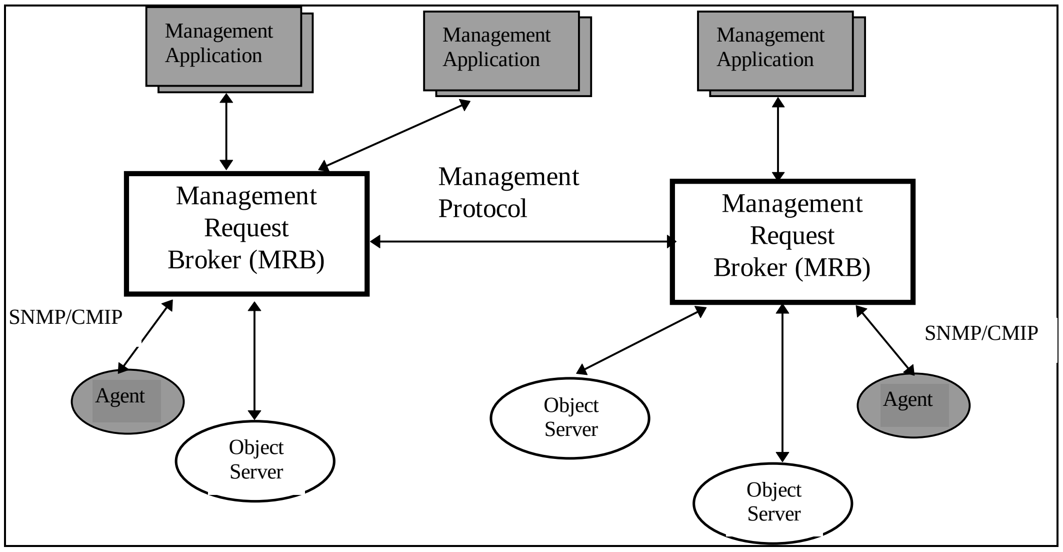

Figure 4. DME in details

a way for management applications to invoke one of the methods associated with a DME object.

The object server provides access to the data stored in the object and can create or delete instances of an object. Whenever the MRB receives a message, it sends it to an object server that is associated with the target object. The MRB finds the object by using the Distributed Computing Environment (DCE) directory services.

The major problem of DME is that, if all network management platform vendors adapt their system to the DME framework, there will be no differentiating features [30]. As a result platform vendors haven't attempted to conform to the it, and although DCE gained success and popularity in the market, DME has not.

4.3 Hierarchical Network Management

Hierarchical network management system [31] uses the concept of SubManager, similar to the manager-to-manager(M2M) protocol described in SNMPv2[32]. A SubManager is associated with a few agents, and collects the primitive information from them, performs some calculations, and produces more meaningful values that can be used by a superior manager. This method significantly reduces the amount of management traffics, because only high-level information is sent to the master manager.

Whenever network operators need more, or high-level, information a downloaded Network Management Procedure (NMP) is dynamically assigned to the system. This allows dynamic reconfiguration at run-time, and removes the need to hard-wire everything at compile time [31]. The NMPs are loaded into submanagers, and are stored in two tables: **`subMgrEntry`** and **`subMgrOps`**. Each procedure is Periodically activated and a "Worker" is created, which

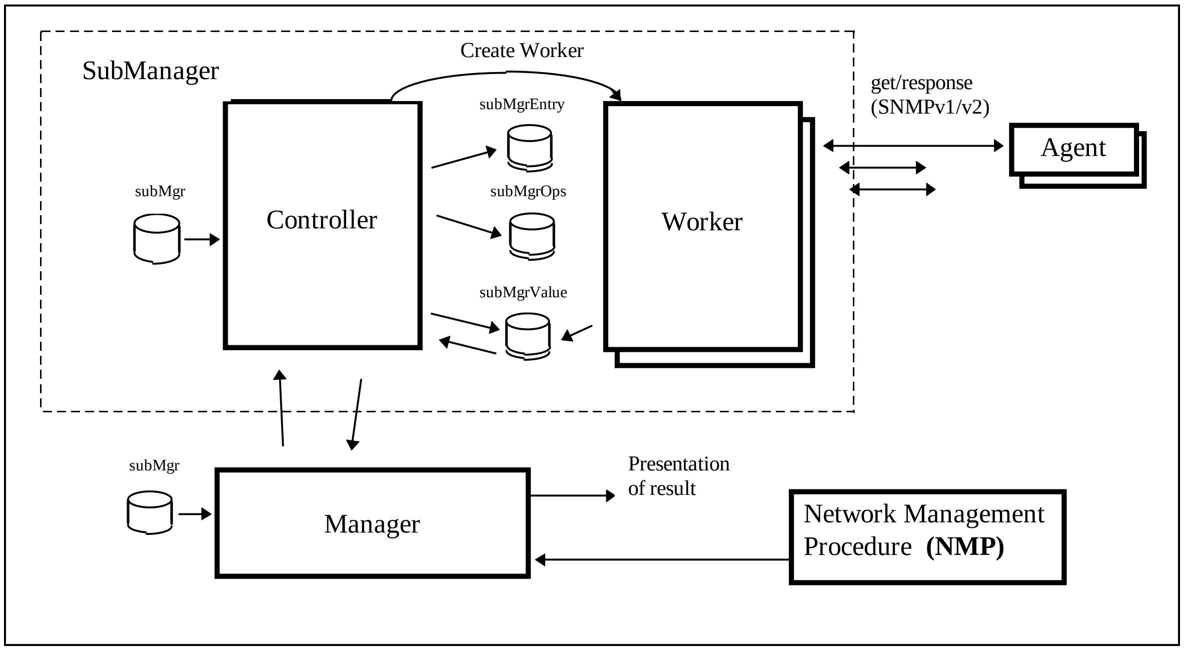

Figure 5. Internals of the SubManager

evaluates the procedure and stores the result into **`subMgrValue`** table This table is read by the manager. This procedure is illustrated in Figure 5.

4.4. Management by Delegation (MbD)

Most distributed management systems use a traditional client/server process interaction method, such as Remote Procedure Call (RPC). The current implementation of this architecture lacks the required flexibility and needs recompilation and reinstallation of the application whenever a modification is made [1]. Management by Delegation (MbD) [33, 34, 35] is a more flexible paradigm and uses the concept of elastic server [36], whose functionality can be extended at execution time by delegating new functional procedures to it.

In MbD, instead of bringing a device's data to the management platform application, applications are delegated to devices and executed in the same environment that data exists. The delegator entity (manager) uses a delegation protocol to download and control a delegated program (DP). An elastic server resides in each device, which maintains and executes DPs, as illustrated in Figure 6.

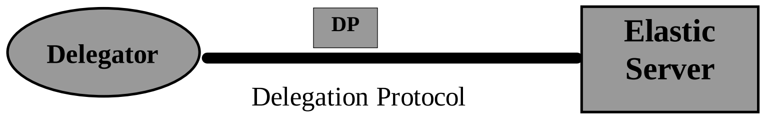

Figure 6. Delegating to an elastic server

A DP may be delegated to a device at boot-time so that the device assumes autonomous responsibilities to handle port failure. This will reduce the need for communication at the times of stress. As automation of management functions distributes the load, the overall load on operation centre is reduced. Also, MbD does not require a uniform semantic model of device data, which simplifies the handling of heterogeneity problem across devices and eliminates barriers to management application development.

MbD can inter-operate with current management protocols, extend their capabilities and even provide a complementary paradigm to them. The entire protocol agents may also be incorporated with an MbD-Kernel as a delegated program, which allows flexible programming for provision of device information. The other feature of delegation is that it may be used as a converter among different management protocol. For instance an SNMP device may be scripted to support CMIP accesses, as well.

MbD improves the survivability of distributed systems by maximising their autonomy. Using MbD allows devices to acquire regional autonomous management capabilities, conditional on the network status. In case of losing communication with management entities, the device may activate management programs that permit fully autonomous management.

5. Comparison

In this section a comparison is made to show the advantages and drawbacks of the discussed systems. We focus our comparison on the performance of these systems for several requirements of distributed network management systems, such as polling method, communication between layers of management, extensibility and flexibility. The comparison is presented in Table 1 and will be explained in the following paragraphs.

	Centralised Management	**Distributed Big Brother**	**Distributed Management Environment**	**Hierarchical Network Management**	**Management by Delegation**
Architecture	Centralised	Hierarchical	Hierarchical	Hierarchical	Distributed/ Hierarchical
Communica-tion method	N/A	Contracting Protocol	RPC	Client/Server	Delegation Protocol
Polling method	Direct	Indirect (via group managers)	Indirect (via object servers)	Indirect (via submanagers)	Indirect (via elastic servers)
Polling Interval	High	Low	Low	Low	Low
Autonomy	N/A	Good	Fair	Fair	Very good
Extensibility	Low	Medium	High	High	Very high
Flexibility	Low	High	Medium	Medium	Very high

Table 1- Comparison of systems

All of these systems share the feature of not polling management information for the whole network from a central location. Instead, they take in events, alarms and alerts from mid-level management platforms. As a result, the number of nodes that management information has to travel is significantly reduced. Although, the communication between each layer of management introduces a new bandwidth penalty, a much less amount of data has to be sent to the higher layer. This is because raw data gathered from network elements is processed before being sent to the higher level. In other words, the management traffic has been localised.

Also, as these systems use parallel polling, they reduce the time to poll the whole network elements' status. Consequently, more recent management information is available. The time to poll the network is directly related to the number of sublayer managers. However, increasing the number of layers increases the traffic because of extra overhead required for manager to submanagers communication. So, there is a trade-off between polling interval and management traffic.

The method of communication between each manager and its submanagers is different for each system. DME and hierarchical network management are based on client/server architecture. DME uses RPC for communication, and hierarchical network management utilises NMPs to download procedure into submanagers. Distributed Big Brother employs contracting protocol to distributed the task between submanagers. MbD, however, employs a completely new approach, known as delegation protocol.

In current RPC-based systems, the communication is limited to the procedures defined at the time of implementation. Although some techniques, such as Dynamic Link Library (DLL), allow more flexibilty, still adding a new functionality to the system might require that the system being shutdown, recompiled and installed again, which cannot be an everyday task. NMP concept improves this situation by allowing the addition of script at run-time. However, MbD enjoys from dynamic reconfiguration by adding procedure at run-time, and delegating them to elastic servers.

Managing a distributed network requires real-time access to the status of each network part. As polling several variables from each segment of the network is not efficient, these data should be compressed. One method of compressing operational data is computation of index functions, known as health functions [1]. Health functions, typically utilise a linear combination of some MIB variables, such as **`ifInOctets`** and **`ifOutOctets`**, at which status indicators change.

Health functions are calculated at lower layers of management from the raw data, and the result is polled by the upper layers from time to time. Whenever a fault occurs in the system, it may be desirable to have a special health indicator which can help the manager localised the fault quickly. In systems other than MbD, the health functions are fixed and can not be changed easily. So, if the required indicator has not been already provided, the manager has to poll raw data which may add a significant amount of traffic.

In terms of autonomy of submanagers, MbD and DBB provide more flexibility. Whenever a fault partitions the network, so that one part of the network becomes unreachable from the central management station, the manager agents in that segment can get full autonomy, and monitor and manage the network, until the connection is restored.

Most commercial distributed network management systems, such as Cabletron's Spectrum, IBM's SystemView and Hewlett-Packard's OpenView, are based on the concept of 'manager of managers (M2M)' [37]. This may be because of simplicity of this approach compared to the others. However, MbD provides a more flexible approach. The popularity of Java [38] as a platform-independent language has eased the implementation of MbD. As trend in network management is toward using flexible and scalable semi-autonomous area managers [39], it seems that MbD will get more popularity than any other available models.

6. Conclusion

In this paper, we discuss several distributed approaches for network management. Four of most important architectures were discussed and compared with each other. Distributed Big Brother is a system based on some AI techniques, such as contracting protocol, and self-organising network. It has been implemented at the University of Michigan, and has shown some satisfactory results when appropriate numbers of LAN and GROUP managers were chosen.

Distributed Management Environment (DME) is based on OSF Distributed Computing Environment (DCE) to bring the management of systems and networks together. DME does not seem to have successed.

Hierarchical network management uses the concept of submanagers and manager of managers of SNMPv2, and utilises the manager-to-manager MIB to communicate between management agents. This idea has been implemented in several commercial systems, because of its simplicity and the popularity of SNMP.

Management by Delegation (MbD) is a new approach for distributed management. It uses the concept of an elastic server, which reside in each management agent, and its functionality can be extended in run-time by delegating procedures to it. It provides significant flexibility and autonomy for management agents. Management of huge and diverse emerging broadband networks will justify the deployment of this approach, and the availability of platform-independent programming language, such as Java, will ease its implementation.

Acknowledgment

This work is supported by a postgraduate scholarship from ministry of culture and higher education of I.R. Iran granted to M. Kahani. The financial support of The Institute for Telecommunication Research (TITR) at the University of Wollongong is also hereby acknowledged.

References

[1] Goldszmidt, German, "On Distributed System Management", In Proceedings of the Third IBM/CAS Conference, Toronto, Canada, October 1993.

[2] Case, J., Fedor, M., Schffstall, M., Davin, J., "A Simple Network Management Protocol (SNMP)", RFC 1157, May 1990.

[3] Meyer, K., Erlinger, M., Betser, J., Sunshine, C., Goldszmidt, G., Yemini, Y., "Decentralising Control and Intelligence in Network Management", Proceedings of International Symposium on Integrated Network Management, May 1995.

[4] Leinwand, A., Fang, K., "Network Management: A Practical Perspective", Addison Wesley, 1993.

[5] So, Y., Durfee, E., "Distributed Big Brother", 8th International Conference on Artificial Intelligence and Applications, 1992, pp. 295-301.

[6] Herman, J., "Enterprise management Vendors Shoot it Out", Data Communication International, November 1990.

[7] Stamatelopoulos, F., Chiotis, T., Maglaris, B., "A Scalable, Platform-Based Architecture for Multiple Domain Network Management",

[8] ITU-T Recommendation M.3010, "Principle and Architecture for the TMN", Geneva, 1992.

[9] Alexander, P., Carpenter, K., "ATM Net Management: A Status report", Data Communication Magazine, September 1995.

[10] Mullender, S., "Distributed Systems", ACM Press Publication, Addison Wisley, 1990.

[11] Spector, A., “Achieving Application Requirement on Distributed System Architecture”, in Distributed System Book, ACM Press, Addison Wisley, 1990.
[12] Coulouris, G., Dollimore, J, Kindberg, T., “Distributed Systems: Concepts and Design”, Addison Wisley, 1994.
[13] Nelson, B., “Remote Procedure Call”, CSL-81-9, Xerox Palo Alto Research Center, 1981.
[14] Weal, W., “Remote Procedure Call”, in Distributed System Book, ACM Press, Addison Wisley, 1990.
[15] “The Common Object Request Broker: Architecture and Specification-Revision 2.0”, Available at URL HTTP://www.omg.org/corbask.htm. July 1995.
[16] Schmidth, D., “Object-Oriented Network programming: An Overview of CORBA”, Available at URL http://www.cs.wustl.edu/~schmidt/corba4.ps.gz.
[17] Agoulmine, N., “Distribution of Management Over Multi-Domains Network Management Systems”, IEEE GLOBECOM’93, 1993, pp. 1217-1221.
[18] RACE Project - Research and Technology Development in Advanced Communications technologies in Europe, RCO-CEC, Brussels.
[19] Ahrens, Mike, “Key Challenges in Distributed management of Broadband Transport Networks”, IEEE Journal on selected areas in communications, vol12, no. 6, August 1994, pp. 991-999.
[20] Kiriha, Y., Nakai, S., Sakauchi, H., Okazaki, F., Okazaki, H., “Concurrent Network Management System using Distributed Processing Techniques”, IEEE GLOBECOM’93, pp. 202-206.
[21] Lee, k., “A Distributed Network Management System”, IEEE GLOBECOM’94, 1994, pp. 548-552.
[22] Stamateopoulos, F., Roussopoulos, N., Maglaris, B., “Using a DBMS for Hierarchical network management”, Engineering Conference NETWORLD+INTEROP ‘95, March 95.
[23] Madruga, E., Tarouco, L., “Fault management Tools for a Cooperative and Decentralised Network Operations Environment”, , IEEE Journal on selected areas in communications, vol. 12, no. 6, August 1994, pp. 1121-1130.
[24] Zhang, X., Seret, D., “Supporting Network Management Through Distributed Directory Service”, , IEEE Journal on selected areas in communications, vol. 12, no. 6, August 1994, pp. 1000-1010.
[25] Smith, R., “The Contract Net Protocol: High-level Communication and control in a distributed Problem Solver”, IEEE Transaction on Computers, C-29(12), December 1980, pp. 1104-1113.
[26] Corkill, D., “A Framework for Organisational Self-Design in Distributed Problem Solving Networks”, PhD thesis, University of Massachusetts, Feb 83.
[27] “DME Overview”, OSF online document, OSF-DME-PD-0394-2, 1992.
[28] Herman, J., “Distributed Network Management: Time runs for mainframe-based systems”, Data Communications Magazine, June 1992, pp. 74-84.
[29] “The OSF Distributed Computing Environment”, OSF online document, OSF-DCE-PD-1090-4, 1992.
[30] Acosta, Victor S., “OSF/DME (Distributed Management Environment)”, Available at URL http://www.frontiernet.net/~/vsa184/paper/osf-dme.htm.
[31] Siegle, M., Trausmuth, G., “Hierarchical Network Management: A Concept and its Prototype in SNMPv2”, JENC6, 1995, pp. 122/1-10.
[32] Case, J, McCloghrie, K, Rose, M, Waldbusser, S, “Introduction to version 2 of the Internet-standard Network management Framework”, RFC 1441, SNMP Research Inc. Highes LAN Systems Inc., Dover Beach Consulting Inc., Carnegie Mellon University, April 1993.
[33] Yemini, Y., Goldszmidt, G., “Network Management by Delegation”, 2nd International symposium of Integrated Network Management, April 1991.
[34] Goldszmidt, G., Yemini, Y., “Evaluating Management Decisions via Delegation”, 3rd International symposium of Integrated Network Management, April 1993.
[35] Goldszmidt, German, “Distributed Management by Delegation”, Proceedings of 15th International Conference on Distributed Computer System, June 1995.
[36] Goldszmidt, German, “Distributed System Management via elastic Servers”, Proceedings of 1st IEEE international Workshop on System Management, April 1993.
[37] Jander, Mary, “Distributed Net Management: In search of Solution”, Data Communication Magazine, February 1996.
[38] “Java Language Specification: version 1.0 Beta”, Sun Microsystem, 1995.
[39] Wellens, C., Auerbach, K., “Towards Useful Management”, The Simple Times, Vol. 4, No. 3, July 1996.