Network Management: State of the Art

Raouf Boutaba and Jin Xiao
Department of Computer Science
University of Waterloo, CANADA
Email: {rboutaba, j2xiao}@bbcr.uwaterloo.ca

Abstract: This paper examines the state-of-the-art enabling technologies for network management, including policy-based network management, distributed object computing, Web-based network management, Java-based network management, code mobility, intelligent agents, active networks, and economic theories. For each of them, we discuss the underlying concept, analyze the benefits and drawbacks, and discuss the applicability to network management. In doing so, we illustrate the common trend in network management design: moving towards distributed intelligence.

1 INTRODUCTION

Nearly a decade ago, the classic agent-manager centralized paradigm was the pervasive network management architecture, exemplified in the OSI reference model, the Simple Network Management Protocol (SNMP) management framework, and the Telecommunications Management Network (TMN) management framework [15]. With the increasing size, management complexity, and service requirement of today's networks, such management paradigm is no longer adequate, and should be replaced with distributed management paradigms. This trend is clearly discussed in [29]. With the myriads of enabling technologies surfaced in the last few years, all of which offering various degrees of network management distribution and benefits, it is unclear what, when, and where are these technologies most applicable? And what would the future of network management be? By examining these state-of-the-art enabling technologies, this paper attempts to shed some light on their benefits, drawbacks, and postulate on their future prospects in network management. Despite their diversity, the paper will illustrate a recurring trend in their design concept: pushing towards distributed intelligence. In a nutshell, management agents are no longer treated as "dumb terminals", but as sophisticated computing devices, and are exploited as such.

Distributed intelligence denotes the management capability and autonomy a management agent exhibits.
This paper is comprised of three parts. First, we will briefly outline the network management concepts, its objectives, and the unique challenges future network management brings. Then we will examine the key enabling technologies for network management. Lastly, we will compare these technologies in terms of distributed intelligence and network resource consumption.

2 NETWORK MANAGEMENT: OBJECTIVES AND CHALLENGES

Hegering [13] defines network management as all measures ensuring the effective and efficient operations of a system within its resources in accordance with corporate goals. To achieve this, network management is tasked with controlling network resources, coordinating network services, monitoring network states, and reporting network status and anomalies. In our view, the objectives of network management are:

- **Managing network resources and services:** including the control, monitor, update, and report of network states, device configurations, and network services.
- **Simplify network management complexity:** it is the task of network management systems to extrapolate network management information into human manageable form. Conversely, network management systems should also have the ability to interpret high-level management objectives.
- **Reliable services:** to provide network with high quality of service, minimize network downtime. Network management systems should detect and fix network faults and errors. And network management must safeguard against all security threats.
- **Cost conscious:** Network management should keep track of network resources and network users. All network resource and service usage should be tracked and reported.

OSI has a well-defined network management reference model [14] pertinent to the designs of current network management architectures.

The OSI model breaks network management functions into the following five functional areas:

- **Fault Management:** the detection, recovery, and documentation of network anomalies and failures.
- **Configuration Management:** record and maintain network configuration, update configuration parameters to ensure normal network operations.
- **Accounting Management:** user management and administration, billing on usage of network resources and services.
- **Performance Management:** provide reliable and high quality network performance. This includes quality of service provisioning and regulating crucial performance parameters such as network throughput, resource utilization, delay, congestion level, and packet loss.
- **Security Management:** provide protection against all security threats to network resources, its services, and data. In addition, ensure user privacy and control user access rights.

In the recent years, network infrastructure is shifting towards service-centric networks. Besides the above network management objectives and OSI functional areas, network management must also fulfill additional management requirements, similar to today's business service models: fast time to market, service differentiation, service customizability, more features, and flexibility.

We envision the future of network infrastructure will drastically change the way network management is done and presents new challenges to network management. First of all, as the size of networks continue to grow at current rate, more and more network devices need to be managed efficiently, demanding better scalability on network management designs. As a result of such size increase, human directives can only be given at a very high level of abstraction and generalization. The underlying network management system must take care of the interpretation of these high-level directives to realizable network configurations and oversee their enforcement. Secondly, as network infrastructures from various sectors converge, heterogeneous network technologies must co-exist and inter-work. Network management systems must provide such seamless integration via common service interfaces, and hide underlying technological heterogeneity from network users. Thirdly, the competitive nature of current network services demands economical operation of networks.

Network management must also be more self-regulating and self-governing, in order to be economically beneficial. At the same time, network management solutions must be kept simple and elegant, as the development of Internet has demonstrated: only simple and elegant solutions would prevail in large-scale heterogeneous networks. Lastly, as network devices become more and more powerful, there is increasing pressure to utilize their processing capabilities. This leads to increasing need for distributed network management at device level.

3 EARLY WORKS TOWARDS DISTRIBUTED NETWORK MANAGEMENT

In the traditional manger-agent network management architecture, such as SNMP, the agent is kept as simple as possible, only tasked with device status report and update, while the burden of management and data processing resides with the manager. Researchers realized the inadequacy of such design around early 90's, as the rapid increase in size of managed network, compounded by increasing demand on network performance and reliability, prompted a complete re-thinking of network management paradigm.

SNMPv2 is the first major installment towards distributed network management. The initial set of Request For Comments (RFCs) (1441-1452) was published in 1992. SNMPv2 introduced the concept of intermediary manager. An intermediary manager can be viewed as a "middle manager". The manager communicates directly with the intermediary managers and exchange command information, while the intermediary managers handle data exchange with agents. In this fashion, the intermediary managers shift some of the data processing from the manager side and is capable of performing simple tasks, such as periodic status pulling from agents, without manager's direct intervention.

In 1995, Internet Engineering Task Force (IETF) took a further step towards management distribution with the proposal on Remote MONitoring (RMON) [38]. RMON used the concept of monitors or probes, which are network traffic monitoring devices. Probe implementation can be done as device embedded applications or as separate devices. The task of a probe is to monitor the network traffic at its local region and report anomalies, in the form of alarms, to its manager. By defining alarm types and alarm thresholds, the manager is able to offload some data gathering and decision-making (mainly

event filtering) to the probes. Furthermore, the probes can also perform some data pre-processing before forwarding them to the manager.
In general, the earlier works towards distributed network management can be considered as weak distribution. The management tasks still reside heavily on the manager side, and some rudimentary management duties are delegated to intermediary entities, in the form of event filtering, notification, and data pre-processing.

4 ENABLING TECHNOLOGIES

We have identified a set of enabling technologies that are commonly recognized to be potential candidates for distributed network management. We will discuss each of them in turn, examine their potential benefits to network management, discuss their drawbacks, and postulate on their prospects. These enabling technologies will be presented in order, with respect to the degree of management capability it bestows on management agents. We believe that distributing intelligence to management agents is an inevitable trend in network management and one that is critical to the success of future network management designs. We will first examine policy based network management. It will be followed by distributed computing, Web-based systems, and Java, which all uses static remote objects to facilitate task offloading from agent to managers. From there, we present the concept of code mobility, in which agents are more management capable, as agents are made mobile and exhibit the ability of independent management processing. A step further in that direction is intelligent agents, where processing units cooperate with each other on peer-to-peer basis, assuming the role of managers and agents interchangeably. Lastly, we will examine the application of active network and economic theories to network management. The former pushes management tasks completely to network devices, and the later forgoes the need for network management infrastructure.

4.1 Policy-based Network Management

Policy-based network management started in early 1990s [30][23]. Although the idea of policies appears even earlier, they were used primarily as representation of information in a specific area of network management: security management [11]. The idea of policy comes quite naturally to any large management structures. In reality, all medium to large size companies today have policies and regulations that their employees must follow. These policies are typically derived

based on company's objectives and goals. In policy-based network management, policies are defined as rules that govern the states and behaviors of the network system. The management system is tasked with: the transformation of human-friendly management goals to syntactical and verifiable rules governing the function and status of the network, the translation of such rules to mechanical and device-dependent configurations, and the distribution and enforcement of these configurations by management entities. The reference model of policy-based network management is largely a manager-agent model, consists of Policy Decision Points (PDPs) and Policy Enforcement Points (PEPs) [18][19]. The first two tasks are handled by the PDPs, while the last task is handled by the PEPs.

IETF's Resource Allocation Protocol (RAP) plays a key role in policy-based network management with its Common Open Policy Services (COPS) [20] and its extension COPS-PR [9]. Some recent works are done on the translation of business directives to network level policies [8] and on policy conflicts resolution [27]. More significantly, the meta-policies concept was proposed in [3]. Its introduction pushes most mundane policy decision tasks from the PDPs to the PEPs. This represents a novel attempt at empowering agents with more management capabilities, moving policy-based network management towards a more distributed intelligence design.

The most significant benefit of policy-based network management is that it promotes the automation of establishing management level objectives over wide-range of network devices. Network administrator would interact with the network by providing high-level abstract policies. Such policies are device independent and human-friendly. The automated translation process will hide the complexity of constructing low-level device-dependent configurations derived from the high-level policies, and therefore facilitate the bridging of business objectives to network configurations. Comparing to human-directed policy translation, such automation would provide more consistent and integrated representation of business objectives. As the state of a network changes, policies would be automatically updated to ensure operational consistency without any human interventions. As today's network increases rapidly in size, such automation is an essential requirement. In contrast to other management technologies, such as Java-based management and mobile agent, policy-based network management allows much more rapid modification of the management requirements after deployment. Policy-based network management can

adapt rapidly to changing management requirements via run-time reconfigurations, rather than re-engineer new object modules for deployment. The introduction of new polices does not invalidate the correct operation of a network, provided the newly introduced polices does not conflict with existing policies. In comparison, a newly engineered object module must be tested thoroughly in order to obtain the same assurance.

For large networks with frequent changes in operational directives, policy-based network management offers an attractive solution, as it can dynamically translate and update high-level business objectives into realizable network configurations. However, one of the key issues in a policy-based network management lies in its functional rigidity. After the development and deployment of a policy-based network management system, the service primitives are defined. By altering management policies and modifying constraints, we have a certain degree of flexibility in cooping with changing management directives. However, we cannot modify or add new management services to the system, unlike mobile code or software agents.

4.2 Distributed Object Computing

Distributed Object Computing (DOC) uses Object-Oriented (OO) methodology to construct distributed applications. Its adaptation to network management is aimed at providing support for distributed network management architecture, integration with existing heterogeneous network management solutions, and provide development tools for distributed network management components. Distributed object computing provides distribution of services and applications in a seamless and location transparent way, by separating object distribution complexity from network management functionality concerns. Another advantage of this separation of concerns is the ability to provide multiple management communication protocols accessed via a generalized Abstract Programming Interface (API), fostering interoperability of heterogeneous network management protocols, such as SNMP for IP networks and Common Management Information Protocol (CMIP) for telecommunication networks. In addition, DOC provides distributed development platform for rapid implementation of robust, unified, and reusable services and applications. Contemporary DOC in network management is oriented around the Object Request Broker (ORB) concept. ORB facilitates

communication between local and remote objects in an effortless way that free the application from low-level infrastructure and communication concerns. The two major adaptation of DOC to network management are: Common Object Request Broker Architecture (CORBA) [32] and Distributed COM (DCOM) [34]. The major application of DOC to network management is mostly in two areas. Firstly, DOC is used to design distributed network management systems, evident in standardization works done by Telecommunication Information Network Architecture Consortium (TINA-C) [33], Joint Inter Domain Management (JIDM) [16], and research projects, such as MESIS [2]. All of these proposed frameworks provide transparent remote services invocation using DOC support. In this fashion, management processing and services need no longer be located at centralized locations in the network, but rather distributed across remote locations. This feature allows management tasks to be delegated, by region or by functional areas, to intermediate entities, making managers no longer the center of all management decision making. Secondly, DOC is used to augment existing network management infrastructures with distributed capability.

Distributed object computing in general, CORBA in particular, is a well-received technology for developing integrated network management architectures with object distribution. The success of CORBA as an enabling network management technology can be attributed to the fact that CORBA has well-established supporting environment for efficient run-time object distribution and a set of support services. In this fashion, DOC is useful as integration tools for heterogeneous network management domains, and extending deployed network management architectures. However, DOC still uses static object distribution. It does not have the flexibility code mobility offers. Furthermore, DOC requires dedicated and heavy run-time support, which may not always be feasible on every device in the network. This later issue restricts its area of deployment.

4.3 Web-based Network Management

Judging by the tremendous success of World Wide Web on the Internet, it is expected that web technology would influence network management to some degree. Today, myriad of web-based network management solutions are proposed and been built, backed up by large corporations, such Sun, Cisco, Microsoft, etc. With respect to network management, the critical problems Web-based network management tries to address are: platform heterogeneity, lack of management console accessibility, and high cost of management

platform deployment and maintenance. Traditional network management solutions are highly platform-dependent. Network administrators must operate on proprietary management consoles to perform daily operations, and the user interfaces for each management platform may vary significantly. Web technology effectively addresses this problem by providing ubiquitous management consoles in the form of standard web browsers. Proprietary network management platforms are expensive and difficult to maintain. Web technology solves this issue by promoting HyperText Markup Language (HTML) and Java applet in information presentation, providing a seamless Graphic User Interface (GUI) accessible everywhere. Lastly, an interesting observation in the IP sector is that network management data is always treated as "second class citizens" compare to user data. While it's true that the transport of management data should never get in the way of transporting user data, the importance of management data is on the rise, especially with the increasing demand on real-time Quality of Service (QoS) services. Using a connection-oriented transport protocol, such as Transport Control Protocol (TCP) for HyperText Transport Protocol (HTTP), implicitly elevates management data to the same level as user data, as viewed by network routers. Web technology serves as a good short-term solution to "patch" the existing issues in network management, as new management paradigms mature, which would take quite sometime to develop and standardize.

We define web infusion as the degree to which web technology is incorporated into a network management platform, ranging from platform extension, to component modification, to full web-based management platforms. In our view, there are three degrees of web infusion existing today: web gateways, web-embedded servers, and web-based management platforms. The web gateways are independent components situated in between web browser type management consoles and management agents, which are implemented as various platform-dependent entities, such as SNMP agents. The web gateway is responsible for the translation of HTTP request to SNMP/CMIP request, and the formulation of web documents based on data gathered from managed devices. The web gateway is extremely easy to deploy, since it does not require any modification on existing management architectures. However, its development can be complex, since it is, by nature, a multi-protocol architectural gateway. In large networks, the presence of web gateways may become performance bottlenecks, as all requests to managed devices have to go through these gateways. The web-embedded servers apply web technology to all managed devices, such as presented in [21] [28]. Each managed device is a miniature web server, capable of accepting HTTP request, processing device data, constructing HTML/XML presentation of device data, and

transmitting constructed documents. Because of the self-contained nature of web-embedded servers, there is no requirement for additional management support. A network administrator can simply interact with a web-embedded device via standard web browser. However, web-embedded servers are not deployable on devices with limited resources and processing power, as it leaves relatively large network footprints. In addition, there are no efficient and economical methods of transforming existing network devices into web-embedded servers. In contrast, Web-based management platforms use web technology as the core technology in the design of new network management platforms, with its own management protocol, data model, and architecture. Web-Base Enterprise Management (WBEM) [37] is a well-known example of web-based management platform. The first two types of web infusion are by far the most adopted solutions in the network management domain today. In both cases, preliminary processing of device data, formulation of status report, and GUI presentation are handled by separate entities other than network managers.

Recently, there has been much debate over the right technology for integrated network management. Since both web technology and CORBA are widely used for this purpose, the question posed comes as no surprise. At first sight, web does seem to be a better choice, as many web advocates believe. Web technology removes the need for proprietary management consoles; it provides uniform management information access via web browsers; data modeling in HTML form is easier than defining Interface Definition Languages (IDLs); with the exception of embedded web servers, web-based management does not need dedicated runtime environment and leaves very small network device footprint; web technology has matured security measures that can be exploited; HTTP based data transport is inherently reliable. However, as we examine the inner works of web technology more closely, the strength of CORBA becomes apparent. Web-based management usually involves much runtime interpretation, in terms of HTML/XML documents, CGI/SSI scripts, and Java applets. These runtime interpretations are a cause of performance concerns, especially for real-time control. HTML/XML are constructed for human readability, hence the formats of these documents tend to be overly wordy for representing key-value pairs, which are the most common type of information in network devices. CORBA’s IDL would be more compact for these types of data representation. And this compactness

translates directly to network bandwidth savings. By using web technology, the developers are limited to using TCP transport for management data, which may or may not be the best choice. CORBA does not place this restriction on the developers. Lastly, CORBA inherently supports distributed management paradigm, by providing support for distributed object development and object distribution transparency. Web technology does not make implementing distributed paradigm in network management any easier. The burden of implementing distribution is largely left to higher-level management architecture. Overall, the choice of technology should be determined based on particular circumstances. In general, web-based technology is better used for providing web access to managed devices, especially if the user of the management application does not have much domain-specific knowledge, e.g. Customer-directed network resource configuration. CORBA is best used for fully distributed network management platforms that values operational efficiency over accessibility. Of course, the two technologies can also be combined in the same management platforms, whereby the web technology could offer access to CORBA-based management applications and services.

4.4 Java-based Network Management

Java, being a portable and object-oriented programming language, is the instrumentation for a wide variety of network management paradigms, ranging from distributed computing, to web-based management, to intelligent agents. Because of this wide applicability, many Java-based development environments have been proposed and designed, supporting network management applications. What makes Java a good technology for network management in general? Firstly, deploying Java-based software solutions are relatively cheap compare to other management software solutions, such as CORBA-based applications. Java virtual machine (JVM) is the only runtime support needed by a Java-based software, and it is also easily deployable and requires very little maintenance. Secondly, as more and more JVM-enabled network devices become available, so does the availability of java support. Furthermore, Java can interoperate with web browsers, which are good candidates for cheap and accessible management consoles. Thirdly, dynamic code downloading allows dynamic distribution of java objects. This not only opens the opportunity for runtime service extensions, but also opens the opportunity for management delegations. Fourth, Java is platform-independent, portable on any existing management platforms that support JVM. Lastly, Java software is easy to develop, as there exists many development supporting environment and tools. Also, Java is a good

programming language for realizing new network management concepts, such as code mobility.

Perhaps the biggest and most mentioned issue with Java is its performance. Java is inherently not an efficient programming language. Besides the obvious performance loss resulting from Java's interpreted nature, Java class loading can be quite slow, especially if dynamic class downloading is required. Java object serialization and remote method invocation are commonly exploited for network management. Both of them have performance problems. Object serialization is quite space consuming, which may not be a big problem on large stations, but would be an issue for small devices. Java Remote Method Invocation (RMI) is not network resource conscious in its operation and tends to waste fair amount of network resources on each method invocation.

4.5 Code Mobility for Network Management

In 1991, Yemini et al. first introduced the concept of Management by Delegation (MbD), and they further refined this concept in 1995 [12]. In their works, Yemini et al. suggested to push management tasks to the agent side. This can be achieved by dynamically transporting programs from managers to agents and perform the delegated management tasks locally. Three immediate advantages of the MbD approach are apparent. Firstly, manager is no longer a centralized processing entity in the network. Much of its processing can be offloaded to agents via delegated programs. Secondly, considerable amount of network resources are saved. For instance, data gathering can be performed locally. Lastly, It is possible to augment the functionality of agents by providing them with delegated programs at runtime. In this fashion, some decision making and network monitoring duties can be performed locally, allowing faster response to management requests and better fault tolerance (in case of manager crash).

Code mobility can be considered as the capability of an application to distribute and relocate its components at run-time. Obviously, there must exist some form of language and run-time support for applications utilizing code mobility. There is much confusion in the literature concerning the terminologies used for code mobility, and the introduction of intelligent agents further blurs the concept. We do not consider intelligent agent as part of the code mobility concept in this

paper. Hence, intelligent agents are considered more complex and self-governed than code mobility, and will be discussed in a separate section. In terms of code mobility, there exist two types: weak mobility and strong mobility. In weak mobility, entire programs or code fragments are transported between distributed components, without retaining execution states and data after transportation. We call applications exhibiting weak mobility as mobile code. Recent works such as [25] explores its use for network management. In strong mobility, the entire program, along with its execution states and data, are transported between remote components. The program will suspend its execution before departure and resumes execution after arrival. We call applications exhibiting strong mobility as mobile agent. Most research works, such as [35][22][26] are focused on this concept. The terms mobile code and mobile agent are often used interchangeably, and sometimes mean different things across literatures.

With code mobility, management tasks no longer have to be performed by the managers. They simply generate management objectives and outline task procedures, the execution of tasks are delegated to the agents. Baldi and Picco [1] defined three code-mobility paradigms based on interaction between services and resources: Code On Demand (COD), Remote Evaluation (REV), and Mobile Agent (MA). In the case of code on demand, the manager has gathered the resources but lacks the code needed for processing. The code is dynamically downloaded from a code server for execution. In the case of remote evaluation, the manager holds the code and the agent holds the resources. The manager dynamically uploads code to the agent side. The uploaded code executes on the resources, and returns back the result to the manager. In the case of mobile agent, the manager holds the services in the form of processing components and the agent holds the resources. The manager relocates the entire processing component, which includes code, execution state, and possibly data, to the agent. If the required data is distributed across a number of different agents, the mobile agent has the ability to relocate from agent to agent, performing data processing and keeping track of generated intermediary data. The MA paradigm is characteristic of strong mobility, while COD and REV paradigms are characteristic of weak mobility.

As mobile code is transported across network, it must be loaded at the destination for execution. The time it takes to suspend execution of a

component, pack its code and data, transport across network, restore the component, and execute, could be quite long. Hence, code mobility is not a good candidate for networks with simple but frequent service requests. Furthermore, to prevent mobile agents from adversely affecting network resources, security measure are often in place which either restrict the operations a mobile agent can perform on local resources, or provide some type of access gateway. Neither solution is satisfactory as access restrictions constrain the operational capacity of mobile agents; while access gateways add unnecessary processing overhead.

4.6 Intelligent Agents

Intelligent agents exhibit the following characteristics: autonomy, social ability, reactivity, pro-activeness, mobility, learning, and beliefs. An intelligent agent is an independent entity capable of performing complex actions and resolving management problems on its own. Unlike code mobility, an intelligent agent does not need to be given task instructions to function, rather just the high-level objectives. The use of intelligent agents completely negates the need for dedicated manager entities, as intelligent agents can perform the network management tasks in a distributed and coordinated fashion, via inter-agent communications. Many researchers believe intelligent agents are the future of network management, since there are quite some significant advantages in using intelligent agents for network management. Firstly, intelligent agents would provide a fully scalable solution to most areas of network management. Hierarchies of intelligent agents could each assume a small task in its local environment and coordinate their efforts globally to achieve some common goal, such as keeping overall network utilization at close to maximum. Secondly, data processing and decision-making are completely distributed, which alleviates management bottlenecks as seen in centralized network management solutions. In addition, the resulting network management system is more robust and fault tolerant, as the malfunction of small number of agents have no significant impact on the overall management function. Thirdly, the entire network management system is autonomous, network administrators would only need to provide service-level directives to the system. Lastly, the intelligent agents are self-configuring, self-managing, and self-motivating. It is ultimately possible to construct a

network management system that's completely self-governed and self-maintained. Such a system would largely ease the burden of network management routines that a network administrator has to currently struggle with.

Wooldridge and Jennings [39] defined three architectural types for intelligent agents: deliberative agents, reactive agents, and hybrid agents. Deliberative agents are based on a physical-symbol system. Such a system describes a physically realizable set of symbols that can be combined to form complex structures. A deliberative agent is able to run processes operating on these symbols to generate overall intelligent actions. Recent works such as [24] make use of deliberative agent. Reactive agents are very much the opposite of deliberative agents. They do not require complex representation of knowledge, nor do they require perfect representation of information. Reactive agents generate behaviors solely based on environmental observations, since they do not include any kind of symbolic world models. In practice, reactive agents are more responsive than deliberative agents due to the lack of any complex symbolic reasoning mechanism. Reactive agents could be successfully applied to traffic monitoring, fault diagnosis, congestion control, and admission control, because these management functions do not have or require perfect representation of a world model. Furthermore, they require rapid responses and actions, which the reactive agents are capable of. Hybrid agents are compositions of both deliberative agents and reactive agents. A hybrid agent would contain a symbolic world model, developing plans, and making decisions in the way a deliberative agent functions. However, it is also capable of reacting to events occurring in the environment without engaging in complex reasoning. The reactive component of a hybrid model overwrites its deliberative component in order to achieve quick response. The hybrid agent seems to be a suitable candidate for fault diagnosis [10]. However, hybrid agents are substantial in size, much larger than either deliberative agents or reactive agents. This may pose a problem when high levels of mobility are expected in a network management system.

The application of intelligent agents to network management is still at its infancy, and much difficult issues still remain unsolved. As applications utilizing intelligent agents arise in network management, the problem of managing these intelligent agents also becomes increasingly important. These self-governing agents cannot simply be allowed to roam around the network freely and access vital resources.

Currently, it is still very difficult to design and develop intelligent agent platforms. This is mostly because very little real-life practices with intelligent agents exist today. We have yet to determine what constitutes a good intelligent agent platform, in practical terms. As more intelligence and capabilities are empowered to the intelligent agents, their size becomes an increasing concern for network transport. Furthermore, agent-to-agent communications typically uses Knowledge Query Manipulation Language (KQML). KQML wastes substantial amount of network resources, as its messages are very bulky. Lastly, protection against malicious intelligent agents is hardly addressed in the current literature. Who takes care of agent authentication? Can agents protect themselves against security attacks? Can agents keep their knowledge secret? How much access rights should agents have over network resources? None of these questions are addressed effectively, and until they do, large deployment of intelligent agents for network management is very unlikely.

4.7 Active Networks

According to Tennenhouse et al. [36], an active network is a new approach to network architecture in which the network nodes, such as routers and switches, perform customized computation on messages flowing through them. In active networks, routers and switches run customized services that are uploaded dynamically from remote code servers or from active packets. The characteristic of activeness is three folds. In device view, a device's services and operatives can be dynamically updated and extended actively at run-time. In network provider view, the entire network resources can be provisioned and customized actively on per customer basis. In network user view, the allocated resources can be configured actively based on user application needs.

Active networks, combined with code mobility, present an effective enabling technology for distributing management tasks to device level. Not only does management tasks can be offloaded to individual network devices, but also the supplier of management task need no longer be manager entities. Such a solution provides full customizability, device-wise, service provider-wise, and user-wise; it provides the means for distributed process across all network devices; it is interoperable across platforms via device-independent active code; it fosters user innovation and user-based service customization; it accelerates new service and network technology deployment, bypassing standardization process and vendor consensus; it allows for

fine grained resource allocation based on individual service characteristics. In the literature, there are two general approaches for realizing active networks: programmable switch approach and capsule approach. Programmable switch approach uses out-of-band channel for code distribution. The transportation of active code is completely separated from regular data traffic. This approach is easier to manage and secure, as the active code is distributed via private and secure channels. It is suited for network administrators configuring network components. On the other hand, the capsule approach packages active code into regular data packets. The active code is sent to active node via regular data channel. This approach allows open customization of user-specified services, however, is more prone to security threats. [4] analyzed the benefits of active networks to enterprise network management.

Quite some recent works are done on exploring active networks for network management, such as the Virtual Active Network (VAN) proposal [7] and the agent-based active network architecture [17]. However, security remains a major roadblock for practical application of active network. Not only the integrity of network resources and user data has to be kept, but also the content of user data must remain confidential. The later implies a strong trust on the active nodes a packet must visit en-route to destination, as it is necessary for user data to be examined and processed in some form. As noted by Murphy et al. [31], there are many objects of security concern in active networks, including: end users, active nodes, Execution Environments (EEs), and active codes. The trust models for these objects are also quite complex. Besides security, resource provisioning and fault tolerance are the other two major issues that need to be addressed in active networks. Firstly, as resources are used for customized processing of data packets in the network. Some means of governing the priority of resource access and the limit of resource consumption has to be established. This issue creates new requirements for network management that must be addressed. Another related issue is network bandwidth consumption. After all, user-specific services must be transported across the network and uploaded. If capsule approach is used, the transportation of these services comes in direct contention with the transportation of user data. Simply charging user for service deployment may not be desirable since it discourages the user from customizing the active nodes in the network. Secondly, fault tolerance of the network will suffer if user-specific services aren’t controlled

properly. As user gains the ability to manage network resources and perform customized processing, more and more user services/applications are injected into the network. The quality of these services/applications cannot be as well ascertained as the manufacturer-supplied services. The obvious solution is to providing each user service with a separate and isolated execution environment. However, such a solution is very costly in terms of resource consumption and network performance.

4.8 Economic Theory

Network management using economic theory proposes to model the network services as an open market model. The resulting network is self-regulating and self-adjusting, without the presence of any formal network management infrastructure. Network administrators would indirectly control the network dynamics by inducing incentives and define aggregate economic policies. Such an approach may seem to be very bold, but it draws its theory from the well-established economic sciences. The premises for applying economic theories are: the existence of open and heterogeneous networks; multi-vendor orientation; and competitive services. Very few works have been done on this subject matter, and most of them are focused on using economic theory as agent coordination model [5][6]. As discussed previously, the management of intelligent agents is still neglected in current literatures. Using economic theory for managing multi-agent systems could be a viable alternative, due to its simplicity and self-sustaining nature.

However, the application of economic theories to network management is only at early experimental stage. Many critical issues brought out with these experiments cast doubts on the applicability of economic theory to network management. Using market model for managing networks is a novel idea. However, some important design issues must be carefully considered. Firstly, the driving force for a market model is the authenticity of its currency. Hence currency values and its transaction processes used in market model must be secure. Furthermore, such secure transactions must be performed very efficiently, as it would be a very frequent operation. Secondly, economic policy for the market model must be designed in such a way that it encourages fair competition, and strongly relates resource contention and its associated price. Lastly, the market model would be

operating on a wide scale, requiring standardization of its elements and operations. Such standardization may be a very slow process and would require full consensus from all participating vendors.

5 CONCLUSION

All of the enabling technologies discussed in this paper attempts to provide distributed intelligence to management agents. Policy-based network management allows managers to partially delegate management tasks to agents in the form of concrete policies. Web-network management offloads the processing, presentation, and display of device information to web gateways or embedded web servers. Distributed object computing, such as CORBA, and Java-based network management provides the means for management task distribution in the network, via deploying static distributed objects. Code mobility and active networks delegate management tasks to management agents through dynamic mobile code downloading. Intelligent agents push distributed intelligence even further by defining autonomous agents that are capable of making complex management decisions. The role of such intelligent agents is no longer confined to either the manager or the agent, as the intelligent agents can adopt these roles dynamically, based on their assigned tasks or their own motivations. Lastly, economic theories completely negate the need for a network management infrastructure, by modeling the network as a self-regulating open market.

To fully leverage the benefits of the presented enabling technologies, network management designers must balance all the benefits and drawbacks, as discussed in this paper. We believe that distributed intelligence is one of the most important trends in the management of current and future large-scale complex networks. Despite the diversity of these enabling technologies, their use in network management research aims at distributing intelligence in the network.

6 REFERENCES

[1]Baldi M., Picco G., Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications, 1998

[2]Bellavista P., Corradi A., Stefanelli C., An Integrated Management Environment for Network Resources and Services. IEEE Journal on Selected Areas in Communcations, Vol. 18, No. 5, May 2000

[3]Boutaba R., Polyrakis A., COPS-PR with Meta-Policy Support, IETF Internet Draft, May 2001.
[4]Boutaba R., Polyrakis A., Projecting Advanced Enterprise Network and Service Management to Active Networks, IEEE Network, Jan./Feb. 2002
[5]Bredin J., Kotz D., Rus D., Economic Markets as a Means of Open Mobile-Agent Systems, In the workshop "Mobile Agents in the Context of Competition and Cooperation" at Autonomous Agents, May 1999
[6]Bredin J., Maheswaran R. T., Imer C., Basar T., Kotz D., Rus D., A Game-Theoretic Formulation of Multi-Agent Resource Allocation, 2000
[7]Brunner M., Active Networks and its Management, NEC-NDLE-IR-2001-5, Feb. 2001
[8]Casassa M., Baldwin A., Goh C., POWER Prototype: Towards Integrated Policy-Based Management, IEEE/IFIP Network Operations and Management Symposium, 2000.
[9]Chan K., Durham D., Gai S., Herzog S., McCloghrie K., Reichmeyer F., Seligson J., Smith A., Yavatkar R., COPS Usage for Policy Provisioning, IETF Internet Draft, draft-ietf-rap-pr-05.txt, Oct. 2000. [RFC 3084]
[10] Cheikhrouhou M. M., Conti P., Labetoulle J., Intelligent Agents in Network Management: A State-of-the-art, 1998
[11] Dobson J.E., McDermid J.A., A Framework for Expressing Models of Security Policy, IEEE Symposium on Security & Privacy, May 1989, Oakland, CA, 1989.
[12] Goldszmidt G., Yemini Y., Distributed Management by Delegation, Proceedings of the 15th International Conference on Distributed Computing Systems, June 1995
[13] Hegering H., Abeck S., Neumair B., Integrated Management of Network Systems, pg.6, Morgan Kaufmann Publishers, Inc. 1999
[14] Hegering H., Abeck S., Neumair B., Integrated Management of Network Systems, pg.82-94, Morgan Kaufmann Publishers, Inc. 1999
[15] Hegering H., Abeck S., Neumair B., Integrated Management of Network Systems, pg.121-152, Morgan Kaufmann Publishers, Inc. 1999
[16] Hegering H., Abeck S., Neumair B., Integrated Management of Network Systems, pg.279-287, Morgan Kaufmann Publishers, Inc. 1999
[17] Hu C., Chen W. E., A Mobile Agent-Based Active Network Architecture, ICPADS 2000
[18] IETF Internet Draft: Policy Framework, draft-ietf-policy-framework-00.txt, work in progress, Sept. 1999.
[19] IETF Internet Draft: Policy Terminology, draft-ietf-policy-terminology-00.txt, work in progress, July 2000.
[20] IETF RFC 2748: The COPS (Common Open Policy Service) Protocol, IETF RFC 2748, Jan. 2000.
[21] Ju H., Choi M., Hong J., EWS-Based Management Application Interface and Integration Mechanisms for Web-Based Element Management, Journal of Network and Systems Management, Vol.9, No.1, 2001
[22] Knight G., Hazemi R., Mobile Agent-Based Management in the INSERT Project, Journal on Network and Systems Management, Vol.7, 1999
[23] Koch T., Kramer B., Rohde G., On a Rule Based Management Architecture, The 2nd International Workshop on Services in Distributed and Networked Environments, IEEE Computer Society, Whistler, Canada, 1995.
[24] Koch F. L., Westphall C. B., Decentralized Network Management Using Distributed Artificial Intelligence, Journal of Network and Systems Management, Vol.9, No.4, Dec. 2001
[25] Lange, D., Java Aglets Application Programming Interface (J-AAPI), IBM white paper, Feb. 1997 (www.trl.ibm.com/aglets/JAAPI-whitepaper.htm)
[26] Liotta A., Pavlou G., Knight G., A Self-Adaptable Agent System for Efficient Information Gathering, 2001

[27] Lupu E., Sloman M., Conflicts in Policy-Based Distributed Systems Management, IEEE Transactions on Software Engineering, Vol. 25, No. 6, Nov. 1999.
[28] Martin-Flatin J., Push vs. Pull in Web-Based Network Management, Technical Report SSC/1998/002, Swiss Federal Institute of Technology Lausanne, 1998
[29] Martin-Flatin J. P., Znaty S., Hubaux J. P., A Survey of Distributed Enterprise Network and Systems Management Pradigms, Journal of Network and Systems Management, Vol.7, No.1, 1999
[30] Moffett J., Sloman M., Policy Hierarchies for Distributed Systems Management, IEEE Journal on Selected Areas in Communication, Vol. 11, No. 9, Dec. 1993.
[31] Murphy S., Lewis E., Puga R., Watson R., Yee R., Strong Security for Active Networks, IEEE OPENARCH 2001
[32] OMG, The Common Object Request Broker: Architecture and Specification, v2.3, Jun. 1999
[33] Prozeller P., TINA and the Software Infrastructure of the Telecom Network of the Future, Journal on Network and System Management, Vol.5, Dec. 1997
[34] Rogerson D., Inside COM, Redmond, WA, Microsoft, 1997
[35] Straber M., Baumann J., Fohl F., Mole – A Java Based Mobile Agent System, 10th European Conference on Object-Oriented Programming ECOOP'96. Jul. 1996
[36] Tennenhouse D. L., Smith J. M., Sincoskie W. D., Wetherall D. J., Minden G. J., A Survey of Active Network Research, IEEE Communications Magazine, Vol. 35, No. 1, Jan. 1997
[37] Thompson J., Web-based Enterprise Management Architecture, IEEE Communications Magazine, Mar. 1998
[38] Waldbusser S., Remote Network Monitoring Management Information Base. RFC 1757, Feb. 1995
[39] Wooldridge M., Jennings N. R., Intelligent Agents: Theory and Practice, The Knowledge Engineering Review. Vol. 10, No.2, 1995