**Literature Survey**

**Introduction**

The more typical traffic scenes used to test a traffic-strategy agent, the more detailed the learning about the advantages and disadvantages of different traffic strategy agents will be. In this case, the initial agent-distribution map will be more accurate. To achieve this superior performance, however, testing a large amount of typical traffic scenes requires enormous computing resources. Researchers have developed many traffic strategies based on AI. Some of them such as neural networks consume a lot of computing resources for training in order to achieve satisfactory performance. However, if a traffic strategy trains on actuator, the actuator’s limited computing power and inconstant traffic scene will damage the performance of the traffic AI agent. As a result, the whole system’s performance will deteriorate. If the traffic AI agent is trained before moving it to the actuator, however, it can better serve the traffic management system. Rational traffic decisions and distributions of agents need the support of ATS, which primarily use agent oriented programming technology. Agents themselves can be humans, vehicles, and so on. To ensure ATS mirrors real urban transportation, we need large computing resources to run many agents.

**Storage**

Vast amounts of traffic data such as the configuration of traffic scenes, regulations, and information of different types of agents in ATS need vast amounts of storage. Similarly, numerous traffic strategy agents and relative information such as control performances about agents under different traffic scenes also consume a lot of storage resources. Finally, the decision-support system requires vast amounts of data about the state of urban transportation.

Two solutions can help fulfill these requirements: Two solutions can help fulfill these requirements:

• Equip all centers of urban-traffic management systems with a supercomputer.

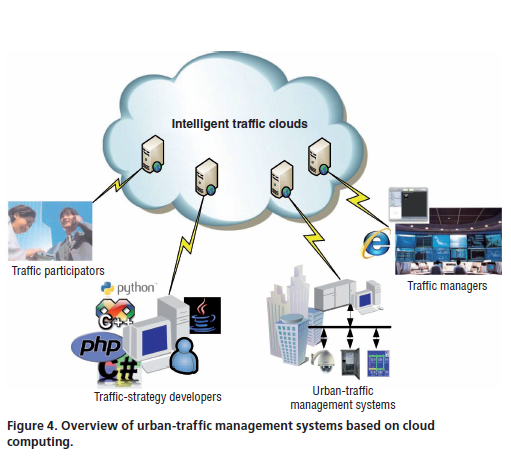
• Use cloud computing technologies

such as Google’s Map-Reduce, IBM’s Blue Cloud, and Amazon’s EC2—to construct intelligent traffic clouds to serve urban transportation. The former both wastes social resources and risks insufficient capacity in the future. On the contrary, the latter takes advantage of the infinite scalability of cloud computing to dynamically satisfy the needs of several urban-traffic systems at the time. This way we can make full use of existing cheap servers and minimize the upfront investment of an entire system.

**Intelligent Traffic Clouds**

We propose urban-traffic management systems using intelligent traffic clouds to overcome the issues we’ve described so far. With the support of cloud computing technologies,

it will go far beyond other multi agent traffic management systems, addressing issues such as infinite system scalability, an appropriate agent management scheme, reducing the upfront investment and risk for users, and minimizing the total cost of ownership.



**Prototype**

Urban-traffic management systems based on cloud computing have two roles: service provider and customer. All the service providers such as the test bed of typical traffic scenes, ATS, traffic strategy database, and traffic strategy agent database are all veiled in the systems’ core: intelligent traffic clouds. The clouds’ customers such as the urban-traffic management systems and traffic participants exist outside the cloud. Fig gives an overview of urban traffic management systems based on cloud computing. The intelligent traffic clouds could provide traffic strategy agents and agent-distribution maps to the traffic management systems, traffic-strategy performance to the traffic-strategy developer, and the state of urban traffic transportation

and the effect of traffic decisions to the traffic managers. It could also deal with different customers’ requests for services such as storage service for traffic data and strategies, mobile traffic-strategy agents, and so on. With the development of intelligent traffic clouds, numerous traffic management systems could connect and share the clouds’ infinite capability, thus saving resources. Moreover, new traffic strategies can be transformed into mobile agents so such systems can continuously improve with the development of transportation science.

According to the basic structure of cloud computing, an intelligent traffic clouds have four architecture layers: application, platform, unified source, and fabric. The *application layer* contains all applications that run in the clouds.

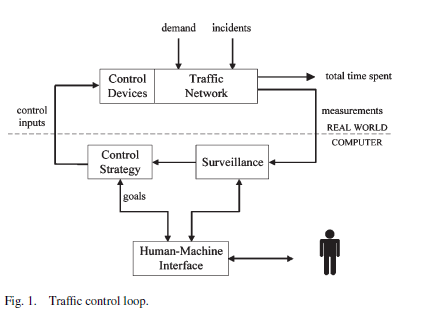
It supports applications such as agent generation, agent management, agent testing, agent optimization, agent oriented task decomposition, and traffic decision support. The clouds provide all the services to customers through a standard interface.

The *platform layer* is made of ATS, provided platform as a service. This layer contains a population synthesizer, weather simulator, path planner, 3D game engine, and so on to provide services to upper traffic applications and agent development.

The *unified source* layer governs the raw hardware level resource in the fabric layer to provide infrastructure as a service. It uses virtualization technologies such as virtual machines to hide the physical characteristics of resources from users to ensure the safety of data and equipment. It also provides a unified access interface for the upper and reasonable distribute computing resources. All those will help solve information silo problems in urban traffic and help fully mine useful information in the traffic data. Lastly, the *fabric layer* contains the raw hardware level resources such as computing, storage, and network resources. The intelligent traffic clouds use these distributed resources to cater the peak demand of urban-traffic management systems, support the running of agents and ATS test beds, and efficiently store traffic strategy agents and their performances

**Neural Networks for Real-Time Traffic Signal Control**

**T**HE INCREASE in urbanization and traffic congestion creates an urgent need to operate our transportation systems with maximum efficiency. As traffic volume continues to increase, the streets become more and more congested. One of the most cost-effective measures for dealing with this problem is traffic signal control. Traffic signal retiming and coordination of existing signals have been proven to bring about substantial reductions in traffic delay, considerable energy savings, and consequently, huge reduction in travel time and increased safety for the public.



Control of traffic signals for efficient movement of traffic on urban streets constitutes a challenging part of an urban traffic control system (UTCS). Traffic signal control varies in complexity, from simple systems that use historical data to determine fixed timing plans, to adaptive signal control, which optimizes timing plans for a network of signals according to traffic conditions in real time. Some of the common traffic signal operations for controlling the traffic flow are cycle time, adjustment; split adjustment, where “split” is defined as the fraction of the cycle time that is allocated to each phase for a set of traffic movements; and offset adjustment, where “offset” is the time difference between the beginning of green phases for a continuous traffic movement at successive intersections that may give rise to a “green wave” along an arterial. The basic elements of a traffic control loop are shown in.

The traffic flow behavior in the network depends on control inputs that are directly related to corresponding control devices, such as traffic lights, variable message signs, etc., and disturbances, whose values cannot be manipulated but may possibly be measurable (e.g., demand) or detectable (e.g., incident). The network’s performance is measured via suitable indices such as the total time spent by all vehicles in the network over a time horizon, the total mean delay experienced by all vehicles in the network, and average vehicle speed. The function of the control strategy module is to specify the control inputs in real time based on available measurements (e.g., from loop detectors), estimations, or predictions so as to achieve the control objectives despite the influence of various disturbances. For UTCS used for controlling traffic signals in a large-scale traffic network, it is crucial that the traffic signal control system has the capability to examine both the microscopic level of the situation (the traffic state of each intersection) as well as the macroscopic level of the situation (the overall traffic state of the traffic network). In addition, the traffic signal control system should be able to adjust various traffic signal control parameters (such as the green time, cycle length, etc.) in response to the varying traffic demand (as opposed to the fixed signal plans of some older systems). However, for a large-scale traffic management system, it may not only be difficult or impossible to tell whether the traffic network is flowing smoothly and assess its current state, but predicting the effects of modifying any of the traffic control parameters is a difficult task due to nonlinear and stochastic events in a traffic network. Additionally, given the complexity and dynamicity of the traffic signal control problem, the control system should be adaptive in nature so that its control function can be adjusted whenever necessary. In fact, such advanced traffic-responsive closed-loop systems and adaptive traffic signal systems are becoming increasingly critical for transportation agencies to meet their day-to-day operation and management needs. It is anticipated that new technology, such as traffic-responsive closed-loop systems or adaptive traffic signal systems using advanced surveillance and traffic management centers, will become increasingly critical for city, region, and state organizations to meet future transportation needs.

The interdependency of each intersection on its neighbors makes it extremely difficult to set the signal parameter values for a large complex traffic network with multiple intersections. An attractive approach to deal with this complicated traffic signal control problem is the use of distributed control technique involving multiple intelligent agents. The primary objective of the multivalent system is to achieve coordinated traffic signal control so as to reduce the likelihood of traffic congestion. It is imperative that such a system performs real-time update of traffic signals in the traffic network based on the changes in the traffic volume. For a case where individual agents are controlling the traffic signals for an indefinite amount of time after they have been installed into the traffic network, the problem of real-time traffic signal control can be said to take the form of an infinite horizon distributed control problem. Hence, for effective traffic signal control, such controllers need to adapt themselves continuously. Various computational intelligence-based approaches have been proposed for designing real-time traffic signal controllers, such as fuzzy sets, genetic algorithm and reinforcement learning, and neural networks (NN). Most of these works are based on the distributed approach, where an agent is assigned to update the traffic signals of a single intersection

based on the traffic flow in all the approaches of that intersection. As some of these models have implemented and tested the controller on a simplified traffic network model consisting of a single intersection, the effectiveness of the proposed neural controller for controlling a large-scale traffic network with multiple intersections cannot be established.

The method proposed in involves the application of simultaneous perturbation stochastic approximation (SPSA) in modeling the weight update process of an NN. Although the SPSA algorithm is a viable option for online weight update as it presents some form of stochastic exploration and converges to a set of optimal values under certain conditions, the model proposed in has some limitations concerning its robustness and responsiveness (more details will be given in Section IV). The work presented by Choy et al. introduced a hybrid multi agent system architecture for real-time system control.

This paper presents an enhanced version of SPSA-NN-based multi agent system, which has been tested in more complex scenarios to determine its efficacy. These two multi agent systems

as well as an existing traffic signal control algorithm Green Link Determining (GLIDE) are used to control the signalized intersections of a large simulated traffic network based on a section of the Central Business District (CBD) of Singapore. This paper seeks to demonstrate the efficacy of the hybrid multivalent system in solving the infinite horizon distributed control problem.

**Cloud Computing and Grid Computing 360-Degree Compared**

**100-Mile Overview**

Cloud Computing is hinting at a future in which we won’t compute on local computers, but on centralized facilities operated by third-party compute and storage utilities. We sure miss the shrink-wrapped software to unwrap and install. Needless to say, this is not a new idea. In fact, back in 1961, computing pioneer John McCarthy predicted that “computation may someday be organized as a public utility”— and went on to speculate how this might occur. In the mid 1990s, the term Grid was coined to describe technologies that would allow consumers to obtain computing power on demand. Ian Foster and others posited that by standardizing the protocols used to request computing power, we could spur the creation of a Computing Grid, analogous in form and utility to the electric power grid. Researchers subsequently developed these ideas in many exciting ways, producing for example large-scale federated systems (Tear Grid, Open Science Grid, cubing, EGEE, Earth System Grid) that defined relevant standards. More prosaically, the term was also co-opted by industry as a marketing term for clusters. But no viable commercial Grid Computing providers emerged, at least not until recently.

So is “Cloud Computing” just a new name for Grid? In information technology, where technology scales by an order of magnitude, and in the process reinvents itself, every five years, there is no straightforward answer to such questions.

**Yes**: the vision is the same—to reduce the cost of computing, increase reliability, and increase flexibility by transforming computers from something that we buy and operate ourselves to something that is operated by a third party.

**But no**: things are different now than they were 10 years ago. We have a new need to analyze massive data, thus motivating greatly increased demand for computing. Having realized the

benefits of moving from mainframes to commodity clusters, we find that those clusters are quite expensive to operate. We have low-cost virtualization. And, above all, we have multiple billions of dollars being spent by the likes of Amazon, Google, and Microsoft to create real commercial large-scale systems containing hundreds of thousands of computers. The prospect of needing only a credit card to get on-demand access to 100,000+ computers in tens of data centers distributed throughout the world—resources that be applied to problems with massive, potentially distributed data, is exciting! So we are operating at a different scale, and operating at these new, more massive scales can demand fundamentally different approaches to tackling problems. It also enables—indeed is only applicable to—entirely new problems.

**Nevertheless, yes**: the problems are mostly the same in Clouds and Grids. There is a common need to be able to manage large facilities; to define methods by which consumers discover, request, and use resources provided by the central facilities; and to implement the often highly parallel computations that execute on those resources. Details differ, but the two are struggling with many of the same issues.

**Defining Cloud Computing**

There is little consensus on how to define the Cloud. We add yet another definition to the already saturated list of definitions for Cloud Computing: A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet. There are a few key points in this definition. First, Cloud

Computing is a specialized distributed computing paradigm; it differs from traditional ones in that 1) it is massively scalable,

2) Can be encapsulated as an abstract entity that delivers different levels of services to customers outside the Cloud,

3) It is driven by economies of scale , and ) the services can be dynamically configured (via virtualization or other approaches) delivered on demand. Governments, research institutes, and industry leaders are rushing to adopt Cloud Computing to solve their ever increasing computing and storage problems arising in the Internet Age. There are three main factors contributing to the and interests in Cloud Computing: 1) rapid decrease in hardware cost and increase in computing power and storage capacity, and the advent of multi-core architecture and modern consisting of hundreds of thousands of cores;

2) The exponentially growing data size in scientific instrumentation/simulation and Internet publishing and archiving; and 3) the wide-spread adoption of Services Computing and Web 2.0 applications.

**Clouds, Grids, and Distributed Systems**

Many discerning readers will immediately notice that our definition of Cloud Computing overlaps with many existing , such as Grid Computing, Utility Computing, Services Computing, and distributed computing in general. We that Cloud Computing not only overlaps with Grid Computing, it is indeed evolved out of Grid Computing and relies on Grid Computing as its backbone and infrastructure support. The evolution has been a result of a shift in focus from an infrastructure that delivers storage and compute resources (such is the case in Grids) to one that is economy based aiming to deliver more abstract resources and services (such is the case in Clouds). As for Utility Computing, it is not a new paradigm of computing infrastructure; rather, it is a business model in which computing resources, such as computation and storage, are packaged as metered services similar to a physical public utility, such as electricity and public switched telephone network. Utility computing is typically implemented using other computing infrastructure (e.g. Grids) with additional accounting and monitoring services.

A Cloud infrastructure can be utilized internally by a company or exposed to the public as utility computing. See Figure 1 for an overview of the relationship between Clouds and other domains that it overlaps with. Web 2.0 covers almost the whole spectrum of service-oriented applications, where Cloud Computing lies at the large-scale side. Supercomputing and Cluster Computing have been more focused on traditional non-service applications. Grid Computing overlaps with all these fields where it is generally considered of lesser scale than supercomputers and Clouds. Distributed computing paradigm or infrastructure that spans across multiple virtual organizations (VO) where each VO can consist of either physically distributed institutions or logically related projects/groups. The goal of such a paradigm is to enable federated resource sharing in dynamic, distributed environments. The approach taken by the de facto standard implementation – The Globes Toolkit, is to build a uniform computing environment from diverse resources by defining standard network protocols and providing middleware mediate access to a wide range of heterogeneous resources. addresses various issues such as security, resource discovery, resource provisioning and management, job scheduling, monitoring, and data management.



Half a decade ago, Ian Foster gave a three point checklist to help define what is, and what is not a Grid: 1) coordinates resources that are not subject to centralized control, 2) uses standard, open, general-purpose protocols and interfaces, and 3) delivers non-trivial qualities of service. Although point 3 holds true for Cloud Computing, neither point 1 nor point 2 is clear that it is the case for today’s Clouds. The vision for Clouds and Grids are similar, details and technologies used may differ, but the two communities are struggling with many of the same issues. This paper strives to compare and contrast Cloud Computing with Grid Computing from various angles and give insights into the essential characteristics of both, with the hope to paint a less cloudy picture of what Clouds are, what kind of applications can Clouds expect to support, and what challenges Clouds are likely to face in the coming years as they gain momentum and adoption. We hope this will help both communities gain deeper understanding of the goals, status, and directions, and provide a more detailed view of both technologies to the general audience.

**Agent-based Network Management System**

Contemporary network management systems as represented by Simple Network Management Protocol (SNMP) are based on the client-server centralized paradigm, where a central station collects and analyses data retrieved from physically distributed network elements. In those systems, management data are stored in a standard structure maintained on the elements to be managed, such as Management Information Base(MIB) Objects Tree in SNMP. There is a daemon agent at each network element, such as sniped running on Linux, which periodically fetches and returns management data in response to inquiries from the manager. Network management system based on Client/Server paradigm normally requires transferring large amount of management data between the manager and agents. The large amount of data not only requires considerable bandwidth, but also can cause a processing bottleneck at the manager. As current networks grow larger and more complicated, the problem becomes more severe.

The solution to such problem is straightforward; distributing the management mechanism to overcome the limitations of the centralized Client/Server architecture.

There are several solutions which have already been put forward, such as Remote Monitoring(RMON) and Management by Delegation(Mad), which introduce some degree of decentralization. A third solution, the use of Mobile Agent(MA) technology to distribute and delegate management tasks has also been investigated Mobile Agent frameworks have already attracted a lot of attention in recent years.

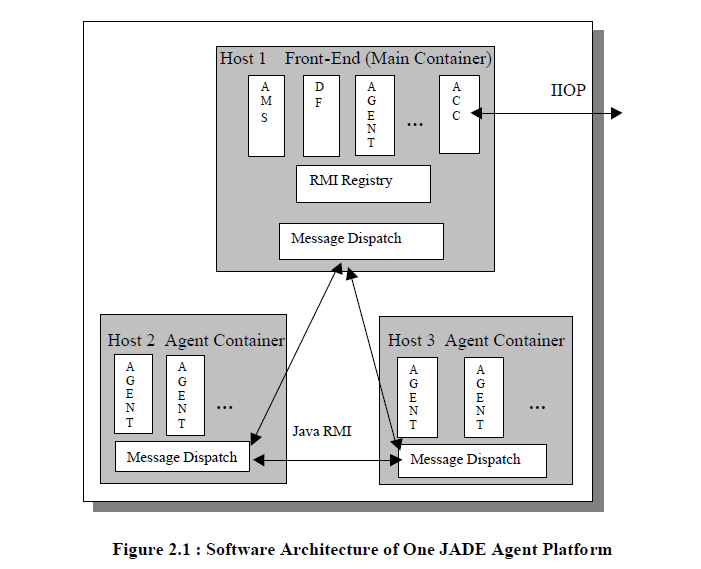
A lot of research is currently being carried out to assess the applicability of agent technology to network management and control environment. It has been argued that the MA have some superior features over SNMP, Mad and RMON . There is a general agreement that MA can be used to alleviate the manager workload and reduce the bandwidth usage by delegation of authority from the manager to MA. MA is more flexible and could be instantly customized by user’s requirement and launched from the manager. It can visit each network element according to the itinerary table, compute and compress the management data locally, only returning the result to the network manager. By moving a portion of the “intelligence” to the nodes where data are resident, many of the management decisions could be taken locally, thus avoiding the transferring of large amounts of data from the remote nodes to the central manager. In order to demonstrate the applicability of MA to network management, we investigated the related problems and implemented a network management system based on mobile agent using the JADE agent platform. Applications implemented in this system are network element’s status monitoring, SNMP table filtering, and global filtering. In this thesis, I will describe the architecture, design and implementation details of those applications. I also investigated the security problem of mobile agent. Security is a crucial problem to the feasibility of mobile agent, especially to the network management domain which has great security requirements. Finally, we tested the performance of the system and analyzed potential problems.

**JADE**

Java Agent Development Environment(JADE) is a software framework to aid the development of agent applications in compliance with the Foundation for Intelligent Physical Agents(FIPA) for inter-operable intelligent multi-agent systems. FIPA is an international non-profit association of companies and organizations sharing the effort to produce specifications for generic agent technologies. JADE complies with FIPA, which includes the Agent Management System(AMS), the default Directory Facilitator(DF) and the Agent Communication Channel(ACC). JADE automatically activates these three agents when the agent platform starts-up. The platform provided by JADE is distributed. It can be split over several hosts with one of them acting as a front end for management and inter-platform communication. A JADE platform comprises one or more agent containers, each living in one Java virtual machine(JVM) and providing the execution environment for the agents. There is one Main Container acting as the front end. It has the supervisory control over the JADE platform. The AMS, the DF and the ACC are in the Main

Container. The general architecture of one JADE platform is shown in Figure 2.1.

Message passing is used to communicate between agents. The FIPA ACL is the language used to represent messages. JADE tries to choose the most efficient way to pass messages between agents as the message protocol.



The goal is to minimize the communication overhead. JADE uses three ways to pass messages:

*1.* **Receiver in the same container of the same platform***:* Java events are used; the cost is a cloning of the ACL Message object and a local method call.

*2.* **Receiver in a different container of the same platform***:* Java RMI is used, the cost is a message serialization on the sender side, a remote method call and a message unserialization on the receiver side.

3. **Receiver on a different platform***:* CORBA IIOP is used; the cost is the conversion of the ACL Message object into a String object and an IIOP marshalling on the sender side, a remote method call and an IIOP unmarshalling followed by ACL parsing on the receiver side. Regarding the agent execution model, JADE uses a *thread-per-agent* concurrency model instead of a *thread-per-behavior* model in order to keep small the number of threads required to run the agent platform. JADE uses Behavior abstraction to model the tasks that an agent is able to perform. Therefore, the agent developer should extend the Agent class and implement the agent-specific tasks through one or more Behavior classes. Finally behaviors are instantiated and added to the agent. In addition, JADE also provide ready-to-be-used library of FIPA interaction protocols.

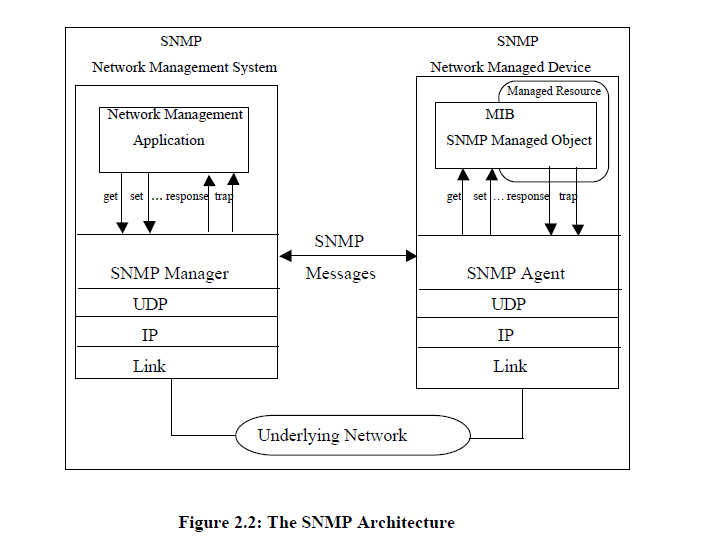
JADE is still an ongoing project, its development and improvement is continuing.

**Network Management**

Network management has been the subject of intense research over the last decade, with the relevant progress being twofold: on the one hand, architectures and algorithms for solving management problems have been devised; on the other hand, different management technologies have been proposed and standardized. Currently, the network management systems adopt a centralized paradigm represented by SNMP. Although it is a standard protocol, it has many limitations and inefficiency. Most reasons are rooted in the centralized architecture. The rational approach to overcome it is to distribute the network management operations. There are several ways which have been under investigation, Management by Delegation (MbD), Remote Monitoring(RMON), using Mobile Agent(MA). The following will give brief overviews of those approaches. The Simple Network Management Protocol(SNMP) was developed in the late 1980’s to provide network operators with a simple tool they could use to manage their networks.

It has gained widespread acceptance since 1993, making it a standard to manage TCP/IP networks, including individual network devices, and devices in aggregate. The more sophisticated Common Management Information Protocol(CMIP) never replaced SNMP. CMIP has only been deployed in the telecommunication networks and not IP networks.

The SNMP Manager makes the connections to a SNMP Agent which executes on a remote network device, and serves information to the manager regarding the device’s status. The database, controlled by SNMP agent, is referred to as the SNMP Management Information Base(MIB), and is a standard set of statistical and control values. Directives, issued by the network manager to a SNMP agent, consist of the identifiers of SNMP variables(referred to as MIB object identifiers or MIB variables) along with instructions to either GET the value for identifier, or SET the identifier to a new value.



Through the use of private MIB variables, SNMP agents can be tailored for a myriad of specific devices, such as network bridges, gateways, and routers. The definitions of MIB variables supported by a particular agent are incorporated in descriptor files, written in Abstract Syntax Notation (ASN.1) format. The popularity of SNMP is due to a number of features. It could cover a large range of devices to be managed, and it is a very flexible and extensible management protocol. It is also proved to be good under poor network conditions. However, SNMP is not a particularly efficient protocol. Bandwidth is wasted with needless information, such as the SNMP version(transmitted in every SNMP message) and multiple length and data descriptors scattered throughout each message. That shortcoming did not stop the widespread use of it.