ARCHITECTURE FOR INTELLIGENT (SMART) AGENTS

NICK PARKYN

Abstract—**Managed nodes, with their associated management agents, are an integral part of any network management environment. Agent technology is moving away from monolithic fixed-function entities to intelligent, flexible agents with dynamic functionality.**

This paper defines common components of agents and defines an architecture for combining these components to build flexible, intelligent agents with scalable performance and functionality. The proposed architecture builds intelligent agents from a number of separate function scalable software "engines".

As a case study, the architecture is used to build a hypothetical intelligent agent for controlling and monitoring network connected devices.

Source of Publication—**OpenView Forum, Seattle, USA, June 1995.**

1 THE TRADITIONAL NETWORK MANAGEMENT MODEL

A network management system contains four components:

- one or more managed nodes, each containing an agent, as shown in Figure 1
- at least one network management station, running one or more network
- management applications (managers)
- a network management protocol, which is used by the station and the agents to exchange management information

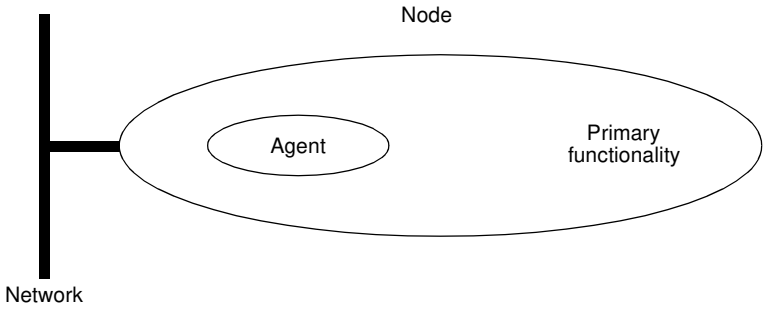

Figure 1: A Managed Node

1.1 MANAGED NODES

The potential diversity of managed nodes is high, spanning the spectrum from mainframes to modems. A successful network management system must take note of this diversity and provide an appropriate scalable framework.

Figure 2 shows how the commonalities between managed nodes as the agent component and can be conceptualised:

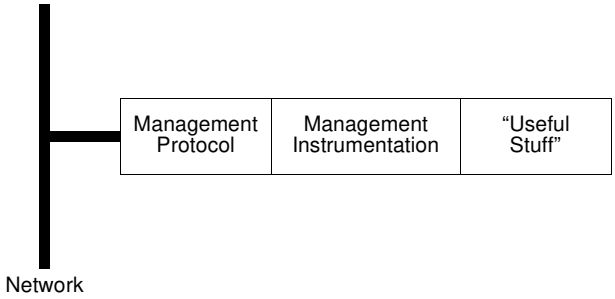

Figure 2: Agent Component of a Managed Node

Each managed node can be seen as containing these components:

- a *management protocol*, which permits the monitoring and control of the managed node
- *management instrumentation*, which interacts with the implementation of the managed node in order to achieve monitoring and control
- *functionality,* "useful stuff" which performs the managed node functions

Traditionally, the interaction between these components has been considered straightforward, with the management instrumentation simply acting as the "glue". The management instrumentation in the managed node implements this "glue" by taking the information available in the node and making it appear as a collection of managed objects. This collection is termed the *Management Information Base (MIB)* held in the agent or the agents object resources.

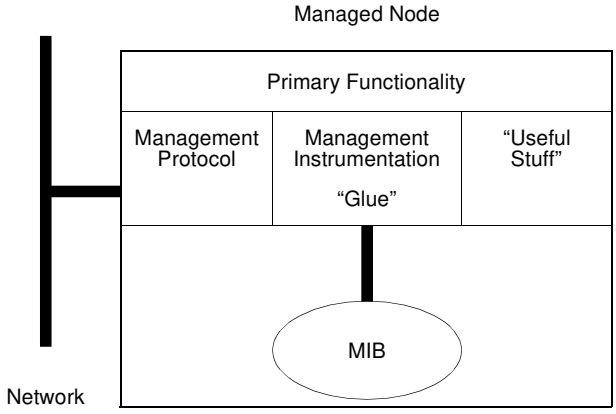

Figure 3: Management Model for Managed Node

Figure 3 depicts the traditional management model for managed nodes, but experience has proven that local processing and decision making with associated action is essential in the agents of devices where:

- real-time or near real-time action is required
- local decisions prevent large volumes of management protocol network traffic
- locally "managed" operation must be maintained even when management applications cannot be reached
- information model transformation

To support this local decision making and action the agent must be "smart" or "intelligent".

2 AN INTELLIGENT AGENT ARCHITECTURE

The fundamental axiom of SNMP-based management is that the impact of adding network management to managed nodes must be minimal, reflecting a lowest common denominator. The concept of intelligent agents does not necessarily undermine this fundamental axiom, provided that we respect the axiom of intelligent (smart) agents as the impact of adding intelligence to managed nodes must be minimal and scaled to requirements and available resources. This is achieved by making the intelligent components scalable, with specific implementations being based on minimum requirements and processing power available at the node.

In defining an architecture for intelligent agents, the management instrumentation is divided into the following physical and logical components:

- "glue" component
- "intelligent" component.

Figure 4 shows the physical and logical components of an intelligent agent.

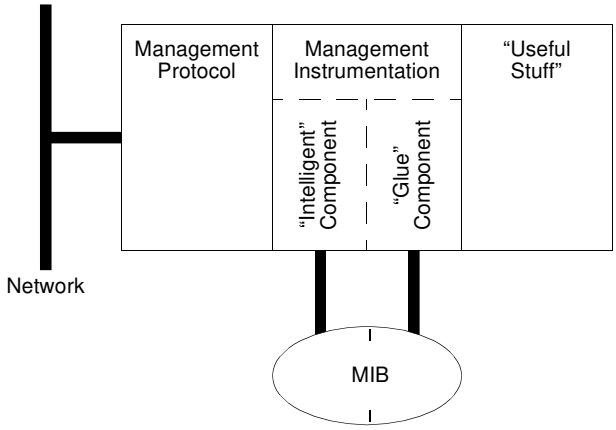

Figure 4: Physical and Logical Component of an Intelligent Agent

The management instrumentation of an intelligent agent comprises:

- "glue" component
- "intelligent" component (software engine)
- set of rules (intelligence)

The "glue" component interfaces to both the "intelligent" component and the management protocol. Both the "glue" component and the "intelligent" components require access to the MIB based object resources and the ability to act upon them.

The MIB can be considered to be divided into two parts with the "glue" component contributing base objects (directly linked to physical environment) and the "intelligent" component contributing higher level objects.

The "glue" component of the MIB provides:

- a fundamental representation of the physical environment and isolates intelligence from the "real world" (enabling remotely modifiable behaviour)
- an interface that does not change unless changes occur in the "real world"
- base MIB objects

The "intelligent" component of the MIB provides:

- programmable functionality
- abstraction
- filtering and correlation
- monitoring
- management actions
- learned behaviour

The scalable technologies used to implement the intelligence could span the spectrum of artificial intelligence (AI) technologies from fuzzy logic to neural networks and beyond, with technology being matched to the requirements and processing power available at the node.

An intelligent agent has the ability to be dynamic by having its set of rules changed or enhanced and by learning "on the job".

3 REDEFINING THE ARCHITECTURE—THE SOFTWARE ENGINE CONCEPT

3.1 REDEFINING THE MANAGEMENT PROTOCOL COMPONENT

While the "front-end" interface of the management protocol is well defined, there has been little or no definition of the management instrumentation to management protocol "back-end".

The intelligent agent concept splits the management instrumentation into two components: the "glue" component and the "intelligent" component. Both of these components have similar requirements in terms of MIB access and management protocol function.

These requirements are:

- to change the value of MIB object (same as SET but a "local" action)
- to interrogate the current value of MIB objects (same as GET but a "local" action)
- to request the Management Protocol to send NOTIFICATIONS.

The management protocol component could be redefined as the *management engine component* and a simple "back-end" interface defined. The management engine component would be responsible for all MIB interactions requested via the front-end or back-end interfaces, with the only access to the MIB being through these interfaces. Figure 5 shows the front-end and back end interfaces and their relationship with the software engine component.

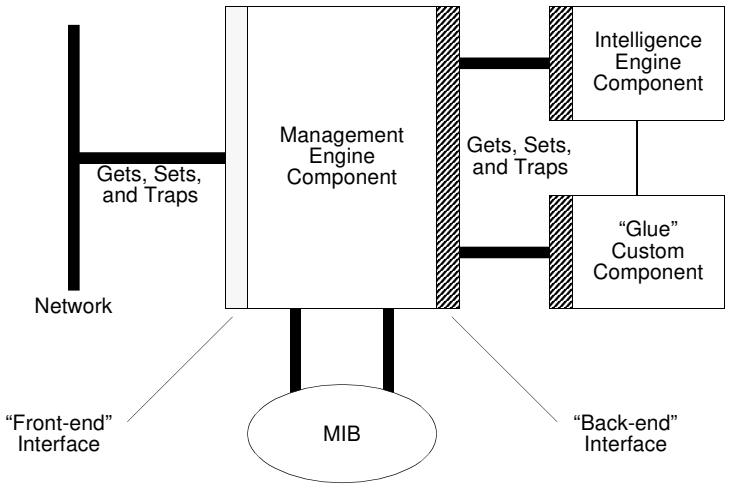

Figure 5: The Front-end and Back-end Interfaces and "Software Engine" Component

3.2 OFF-THE-SHELF SCALABLE FUNCTIONALITY

Redefinition of the architecture has resulted in components with clearly separate functionality and defined interfaces. Both the *management engine* component and the *intelligence engine* component could be considered as scalable "plug-in" "software engine" entities. The custom "glue" component is the only part which is specific to the hardware and functionality of the particular node. This presents possibilities for "off-

the-shelf" selectable and scalable functionality. Intelligence can be downloaded in the form of rule sets to provide flexibility and extensibility. Figure 6 shows a scalable plug-in software engine.

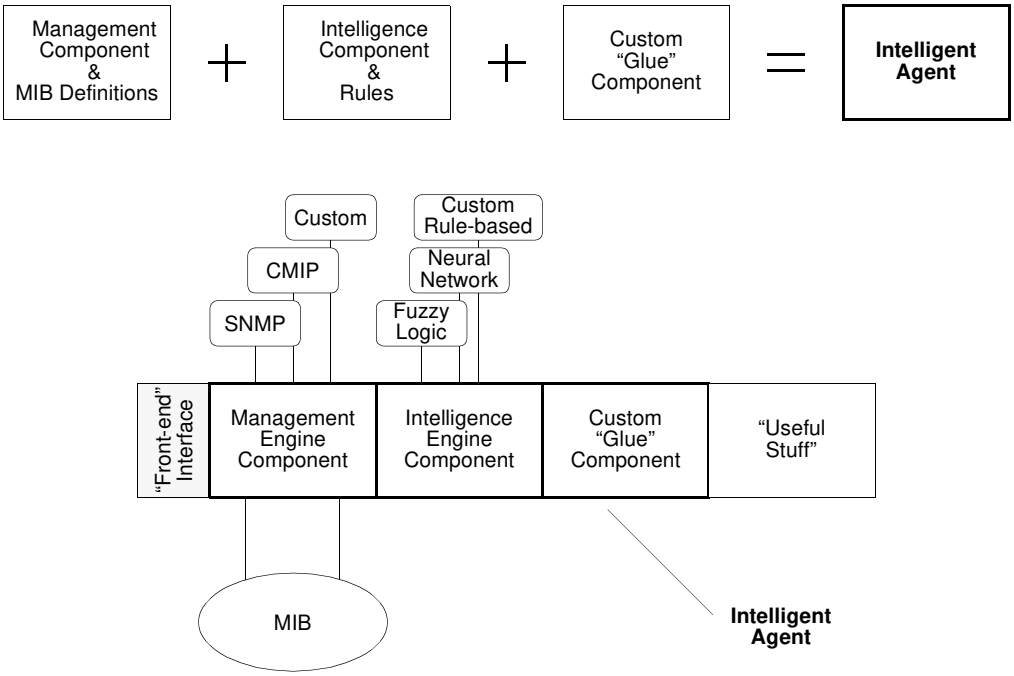

Figure 6: A Scalable "Plug-in" Software Engine Concept

3.3 OBJECT-BASED IMPLEMENTATIONS

This redefined architecture is better suited to object-oriented implementations with the management engine being the object manager of a collection of managed (MIB) objects. Figure 7 shows an object based implementation of an intelligent agent.

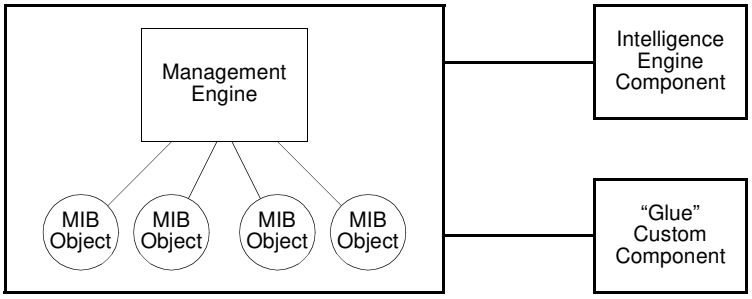

Figure 7: An Object Based Implementation of an Intelligent Agent

4 A SIMPLE CASE STUDY

Let us consider a hypothetical case where a managed node controls a number of devices. The requirements are that this managed node:

- control a number of connected devices, each with a number of discrete signals which must be monitored
- use SNMP management protocols
- have local intelligence
- run on low cost hardware with limited processing power

Figure 8 depicts this scenario:

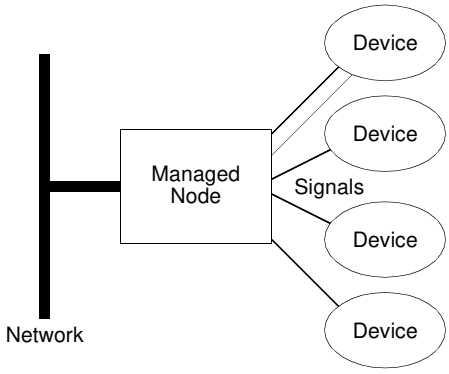

Figure 8: Managed Node Controlling Devices

There is a requirement for the association of more than one signal with a particular device error or device operating condition. Since we have limited processing power we would be wise to choose a rule-based technology with low overhead. Fuzzy logic is well suited to device control and requires little processor power. The SNMP management protocol (by design) has low processing requirements and is a specified requirement.

The custom "glue" component is custom code which scans the hardware ports to which the devices are connected and update MIB variables based on their current state.

The MIB definitions model signals which are associated with particular port bit positions. They are "base objects" but can be combined to map "compound" conditions.

Fuzzy logic rules associate inputs and states with higher level "compound" conditions.

Based on our architectural model we would expect to use an "off-the-shelf" SNMP management engine and fuzzy logic intelligence engine components. We code the custom "glue" component, define the MIB objects and define the fuzzy logic. Figure 9 depicts this combination.

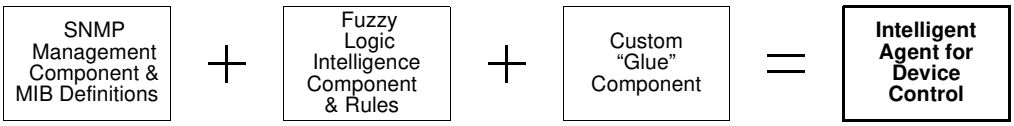

Figure 9: Components of an Intelligent Agent Implementation

4.1 FUTURE EVOLUTION

Driven by the requirement for efficient distributed functionality, logical functionality could be distributed in special purpose "intelligence", database and "agent" servers. Also, the ability to ensure the update of a number of objects in a single operation and the grouping of objects into secure domains is an essential requirement of high-end network management systems. This requires transaction processing (TP), which guarantees update of all specified objects or provides rollback to previous values and secure communication. The concept of servers running multiple agents in a secure domain would support these concepts.

Communication with the multi-agent server is implemented using traditional network management protocols like SNMP, CMIP, and CMIP with transaction processing, while server-to- server communications is should this be future tense? implemented using client-server protocols, which provides transaction processing capabilities.

MIB "database servers" and "intelligence servers" would allow integrated interaction between the objects of managed devices in the network. Figure 10 shows this multi-agent server architecture:

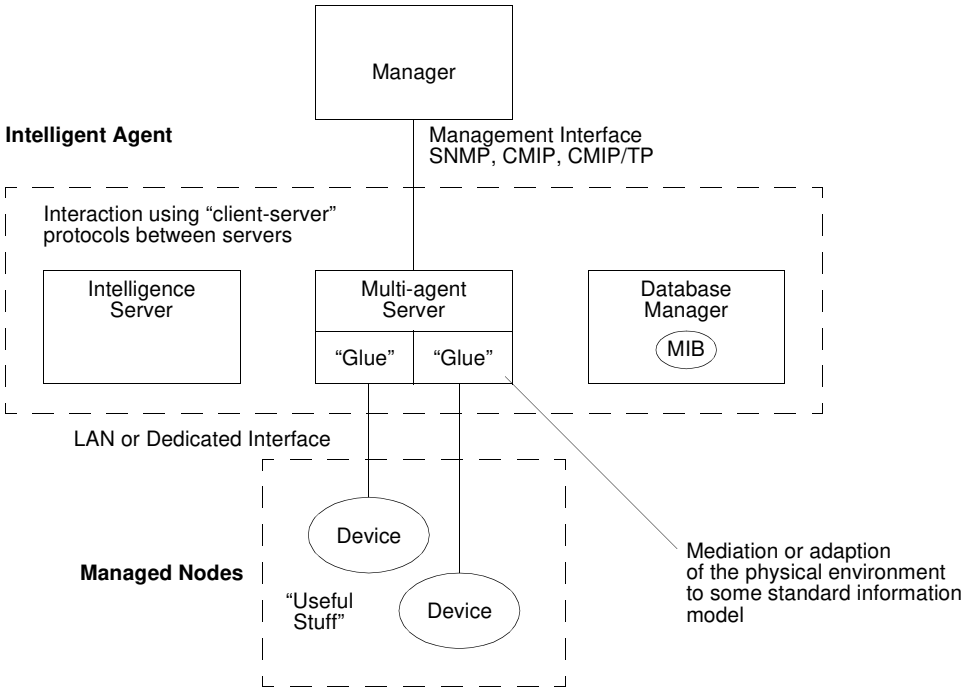

Figure 10: Multi-agent Server Architecture

The advantages of the multi-agent specialised server topology are that:

- Low-level devices could use simple bus extension protocols allowing management of devices which could not even support the overhead of more complex protocols like SNMP.
- There is less duplication of network management protocol code (not required in each device).
- Servers are optimised for a specific function and have the processing power to implement complex client-server protocols with transaction processing.
- Redundancy can be supported at server level.

5 CONCLUSION

Agent technology is evolving from monolithic fixed-function entities to intelligent, flexible agents with dynamic functionality. This functionality can be achieved with incremental change to existing architectures and without even undermining the fundamental axiom of SNMP, provided that we respect the axiom of intelligent agents— that the impact of adding intelligence to managed nodes must be minimal and scaled to requirements and available resources.

There are some clear paths for future evolution moving away from the current agent architecture towards special purpose "software engine" based servers and "multi - agent servers" based on distributed client-server models.