

Increasing the Reliability of Control Systems with Agent Technology

Siritorn Prayurachatuporn

King Mongkut's University of Technology, Thonburi, Bangkok, Thailand
and

Luigi Benedicenti

University of Regina, 3737 Wascana Pky, Regina SK S4S 0A2, Canada

Abstract

This paper describes how Software Agents technology can be used to increase the reliability of a control system. Supervisory Control and Data Acquisition (SCADA) systems are established as means to exercise supervisory control over an industrial process. Software Agent technology can improve SCADA systems as it allows distribution, which inherently promotes redundancy, and modularity, which promotes versatility.

1. Introduction

In the last few years, distributed computation has been the topic of many dissertations and research papers [2-6]. The interest in distributed computation has resulted in a number of enhancements in networking, leading to the construction of internetworking protocols and the creation of a large geographic network (Internet) to implement them [1, 5]. This paper presents how the development of a directory service protocol using Software Agent technology increased the theoretical reliability of Supervisory Control and Data Acquisition Systems (SCADA).

Software agents are autonomous program units supported by an execution environment; software agents can use the network to send themselves to other processors, thus "moving" among computers. Moreover, software agents can talk to each other on the network. This allows them to be used as elementary building blocks for complex systems.

Control systems can benefit from agent technology in many ways. First of all, modularity is the key requirement of a control system and a key property of an agent system, so a control system can benefit from the already existing agent structures.

Second, autonomy is another characteristic of agents that allows the user to accomplish the tasks in a collaborative manner with the computer. Thus, instead of exercising complete control and taking responsibility for every move the computer makes, people can engage in a cooperative process in which both human and computer agents initiate communication, monitor events, and perform tasks to meet a user's goals.

Third, through pro-activity agents take initiative and change their environment. For example, once an agent has completed its task on a machine or is unable to do it, the agent migrates.

This paper is organized as follows. Section 2 presents an overview of elementary SCADA component reliability. Section 3 presents an overview of agent systems and of an agent-based architecture for SCADA

systems. Section 4 presents an analysis of agent-based SCADA systems. Section 5 concludes the paper.

2. Reliability analysis

The reliability of a SCADA system depends on its components. It will be assumed that n different components are needed to make the system work. Moreover, each component's reliability can be considered independent of every other's.

A common reliability model for this kind of system is based on the joint distribution of n random variables. Since they are independent, their probability distribution can be calculated as the integral of the combination of the individual distribution functions.

If the system fails if any one of the components fails, then its reliability is given by the following expression:

$$R_z[t < t_0] = 1 - F_z[t < t_0] \quad (1)$$

Where $F_z[t < t_0]$ is the probability that the system fails before the time t_0 . Furthermore, since this probability results from a series of independent probabilities, it is admissible to write:

$$F_z[t < t_0] = f_{x1}[t < t_0] \cdot \dots \cdot f_{xn}[t < t_0] \quad (2)$$

In the most trivial case, one can assume a binomial distribution for the cumulative probability of failure at any given time, thus the probability of failure will be higher than the probability of any one system failing.

This is a direct derivation of the fact that the components are connected in series. Moreover, any redundancy can be translated in a parallel for each component, thus reducing the probability of failure for the component (ideally halving it) but unfortunately any such redundancy can only be local. So to make the full system redundant it would be necessary to double the amount of components.

Of course, the ideal solution to this problem would be a "joker" component, capable of assuming the form of any other component, and thus effectively being able to replace any of them. This would double the system's reliability with the minimum expense. This paper shows how an agent-based architecture succeeds in this task.

3. SCADA infrastructure

A conventional SCADA system is based upon a collection of functional block that realize dedicated functionality such as data acquisition, decision support, database updates, etc (Figure 1).

This realization of a SCADA system suffers from the shortcomings presented in the previous section, as its reliability is fundamentally connected to the well being of every component it is comprised of. Although for low-level components (e.g., PLCs) it might be possible to

conceive redundancies, it is costly to replace dedicated servers as each needs different treatments.

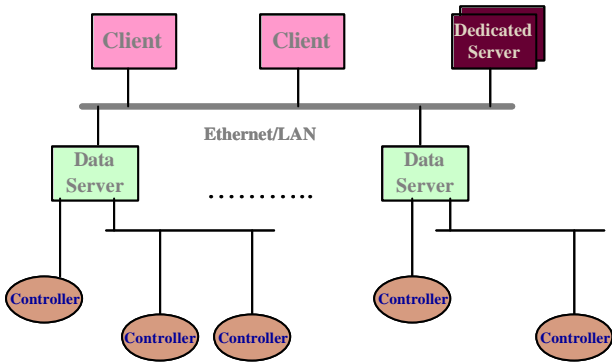


Figure 1: Conventional SCADA architecture

Applying an agent execution environment (AEE) to a real-time distributed control systems means to develop a new protocol for a distributed control system that utilizes mobile agents. An agent execution environment is a software system that provides a runtime environment for agents to execute, a standard interface for interactions, services for creation, migration and termination of mobile agents, supports agent mobility and communication while providing security for both hosts and agents.

The main purpose of the AEE is to host agents and allow them to run. Primarily, an AEE is composed of two main types of agents: system agents and user defined agents. System agents are created by the AEE and are used to help the AEE operate through the use of their services. On the other hand, user defined agents are agents which use the services of the AEE in order to operate.

A mobile agent system consists of one or more AEE linked together. An agent execution environment will consist of many services that facilitate and control mobile agents.

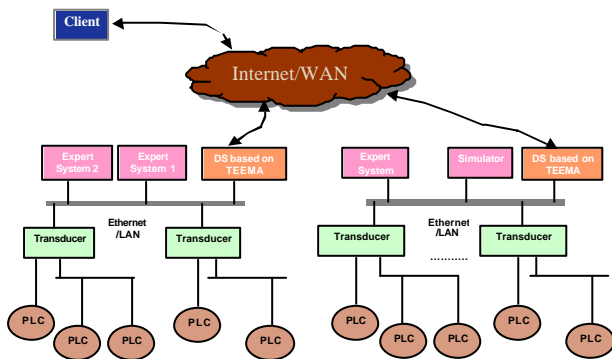


Figure 2: Agent-based SCADA architecture

The application of an agent-based infrastructure and architecture to a SCADA system is relatively straightforward in principle. However, some design issues need to be addressed at the very beginning of the characterization. First of all, it is necessary to identify the variation points of the SCADA system architecture, given an agent-based environment. There are three main features of an agent-based system that are advantageous for a SCADA system: modularity, communications, and mobility.

The protocol of the directory service system is mainly used to locate agents in the linked agent execution environments, or to give advice for the existing agents to other agents.

There are two types of directory service: the local directory service agent and the central directory service agent. Each directory service is given a unique name in the agent execution environment during its initialization. This ensures location-transparent names at the application level. Thus, every service knows how to find the directory service itself.

On the other hand, agents must be identified uniquely in the environment in which they operate. Proper identification allows control, communication, cooperation, and coordination of agents to take place.

A SCADA system takes advantage of many features provided by an agent execution environment. Above all, the most important feature is location independence. With it, an agent is no longer limited on a single machine, but it can be accessed wherever it is located. This makes it possible to implement a failsafe mechanism, in which multiple copies of an agent may be activated in different locations, thus effectively improving the reliability of the system.

Another important advantage of agent systems is platform independence. While this comes at a cost (mostly speed) it is of great help for components that now can work on any workstation equipped with the agent execution environment. The agent execution environment thus makes the workstation aware of other workstations, thus realizing a more sophisticated form of a network: a cluster of awareness (Figure 3).

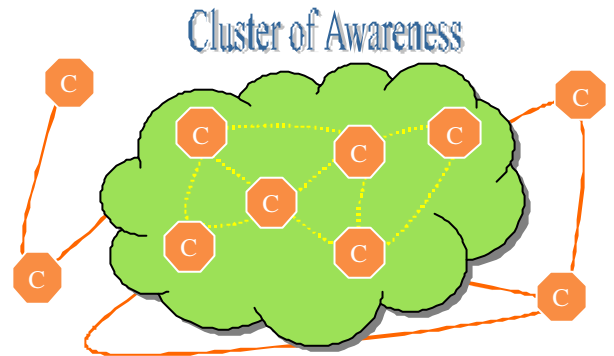


Figure 3: A cluster of awareness

An agent execution environment changes the underlying structure of the SCADA system, but not its conceptual architecture. In fact, the adoption of an agent execution environment streamlines the creation of modules and promotes a distributed model without having the need to make changes to the SCADA architecture itself. However, since the agent paradigm requires substantial changes in the program code, it can be very costly to extend a SCADA system for software agents, and it is also not feasible to rewrite the entire system overnight.

Evolutionary migration patterns can be used that transform the existing SCADA system into an agent system by gradually replacing components with their agent counterparts, while at the same time providing stub interfaces for the non-agent components.

4. Implementation analysis

The most important part of an agent system that implements a directory service is, of course the directory service itself. The work described in this paper has been applied to the TEEMA execution environment. TEEMA stands for TRILabs Execution Environment for Software Agents. TEEMA is a lightweight development environment developed at TRILabs, a western Canadian partnership involving western Canadian provincial governments, universities, and companies. TRILabs's goal is to foster pre-competitive research endeavors, and TEEMA represents one such result.

TEEMA was built for fast agent deployment, in order for researchers to study the characteristics of agent-based programming without being constrained by code unavailability, as is often the case for commercial systems. TEEMA is very modular, and as such it is very change-resistant. Although TEEMA has not been tested in industrial environments, nonetheless any result obtainable in such a "bare-bones" environment can be replicated easily in a more robust system.

The directory service in TEEMA is an agent itself and as such it has no special privileges, other than the possibility to specify its name. Figure 4 shows the block diagram of a system using the services provided by the directory service. The directory service communicates to other subsystems using a standard protocol.

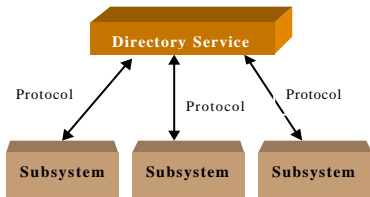


Figure 4 Block diagram of the directory service system.

The general model of a service agent is shown in Figure 5. The model for the agent is that of an event-driven agent with storage. This allows the agent to provide a service that can either maintain a state or not. It is important to notice that this agent does not have any reference to the protocol that it employs to communicate with the agents requesting its services; it must comply, however, with specific requirements when interacting with the directory service.

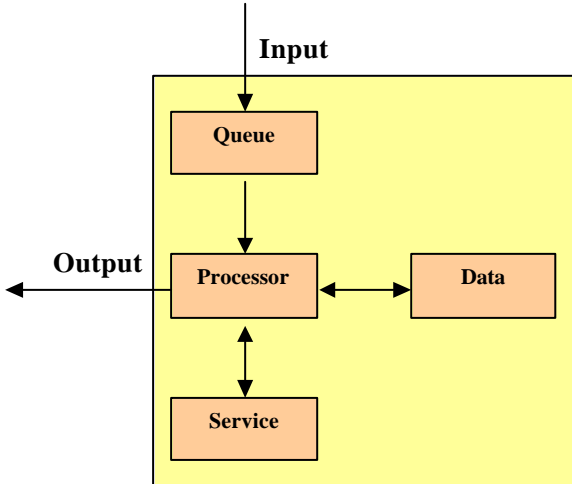


Figure 5: General block model of a service agent

The directory service system prototype is shown in Figure 6. Each directory service agent maintains three collections. The first one keeps track of the service agent's name and agent address. The second one tracks non-local services names and addresses. The last one is only maintained by the central directory service: it is a Directory Service Name table that is composed of the directory service name and the directory service agent address, to locate where the other directory service agents reside (Figure 7).

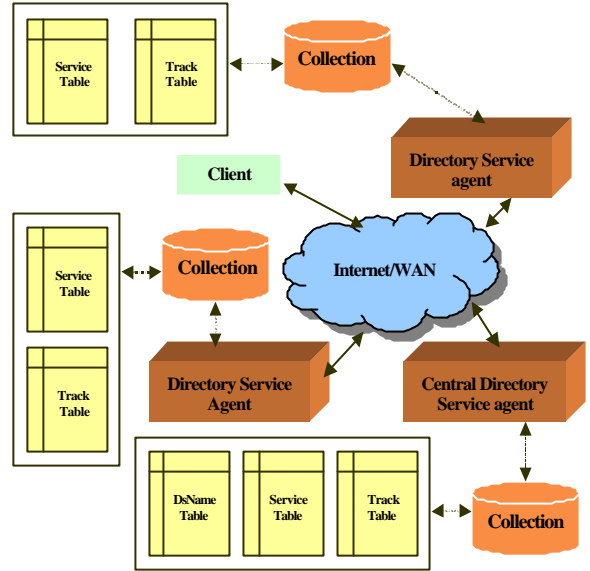


Figure 6 Prototype of the Directory Service System.

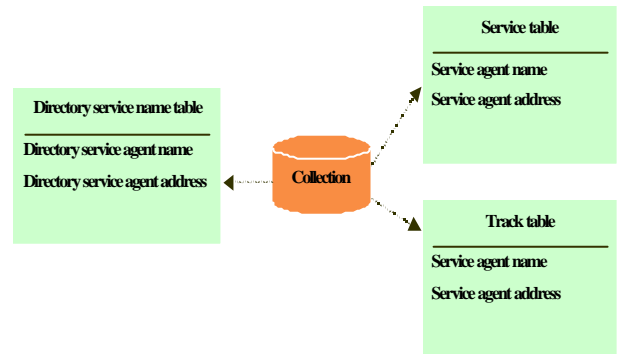


Figure 7 Directory Service Agent Collections.

As a general rule, each agent execution environment should have one or more than one agent running at the same time, and each agent is usually assigned specific tasks to complete. Agents can all be aware of and communicate with each other, even when they are in different machines.

The directory service system is a service provided by agents, running as a system agent. All user-defined agents that are used in an agent execution environment are subclasses of the base agent class and are started by the user. The directory service system in an agent execution environment should provide the following services:

- **Creation of the directory service agent:** The directory service agent is running as a part of the system agent pool. It should be created when an agent execution environment starts. In the prototypical implementation presented in this paper,

it is incumbent on the user to start it as “The Local Directory Service Agent” or “The Central Directory Service Agent.”

- **Sign in:** Every service agent that is created in an agent execution environment will register itself to the directory service agent with two parameters: service agent name and service agent address.

The directory service agent also signs in to the central directory service agent for which the transaction occurs between the local directory service agent and the central directory service agent. This ensures location independence.

- **Sign off:** Services must register with the directory service when they are created, but they must also let the directory server know when they wish to sign off. This is done to update the table of services for each agent platform; naturally, it may happen that some of the agents experience problems and as such they are unable to respond. This is implemented through the use of “probe” messages. If messages cannot be delivered, then the services become unavailable, and automatic sign-off is enforced. If messages can be delivered, the a long timeout is considered.

In exactly the same way, the directory service signs off from the central directory service when it is stopped running the system.

- **Query:** Each agent can query the directory service for other service agents in both the same and different locations. First, the directory service agent queries database to find out whether a requested service is in its local environment or not. If it is, it will give the result to the requesting agent. On the other hand, if the requested service is not in the local environment, the directory service will forward the query to the central directory service agent and wait for the result within a specified amount of time (long timeout).

The central directory agent can broadcast a query to other local directory services, once it receives a query from another directory service agent and cannot find it in its environment. The central directory service agent will also broadcast the request to other directory service agents in case its table does not contain a service being requested.

- **Cache:** Each transaction in an agent execution environment will be kept in the tracking collection. For example, if an agent requests a service which it is not in its local environment, the local directory service will query the central directory service and when it becomes known where the requested agent is, the local directory service will keep the agent name and its address into the collection. If the same service is asked to the directory service again, the directory service can extract its location from the track table directly. It is also very important to update these records often. It is possible that a specific service migrates from one platform to the other, but this is taken care of by the agent execution environment. Instead, if a service agent ceases to exist in a non-graceful way, it is important to have a recovery mechanism (such as the one described in the previous paragraph) to ensure the table consistency.

An analysis of these processes yields that there are two kinds of channel that are used to communicate within the environment with the directory service system: the Internal Channel and the External Channel. The Internal Channel is used for any transaction among agents and the directory service in the agent execution environment. For example, after the directory service agent receives a request, it queries whether the requested agent is in its environment or not. If it is, the directory service agent sends a results message to the requesting agent. Afterward, the two agents are able to communicate with each other.

The External Channel is used in order to communicate among directory service agents. From the above example, the directory service agent, using the internal channel, acts as the “Local directory service agent.” If it cannot find the requested agent in its environment, it forwards the request command to the central directory service using the external channel, and then it waits for a response (within a certain amount of time).

The implementation of the elementary operations results in a protocol exemplified in Figure 8. The figure shows the registration procedure (a), the local query using the internal channel (b), the remote query using an external channel (c), and the results of the query (d).

4.1 Advantages

There are many advantages of applying the agent technology to a distributed control system, as follows:

- **Integration**

A SCADA system implementing an agent architecture is much easier to integrate than their non-agent counterpart. This simple result derives directly from the characteristics of the agent architecture. Agents are intrinsically modular and they communicate through a well-established mechanism. Thus, they can be implemented in a straightforward way, and their structure can become more repetitive, which promotes reuse and quality checking.

- **Reliability**

Reliability is the primary concern of this paper, and since the agent execution environment can transform a monolithic SCADA system into a parallel system, the reliability of an agent-based SCADA is greatly enhanced. Furthermore, it is possible to have a reliability service whose sole task is to monitor the existing services and make them available to other parts of the SCADA system. This system recovery tool makes the system more fault-tolerant.

- **Versatility**

The usage of mobile agents over the Internet improves the versatility of the SCADA system. Not only each system is reliable and fault-tolerant in their site, but they are also able to access other remote systems if the need arises (local system partial failure)

- **Remote access**

The network infrastructure and the inherent distributed nature of the agent architecture make it possible to control the SCADA system remotely, for example, operators can control their plants from remote sites via the Internet.

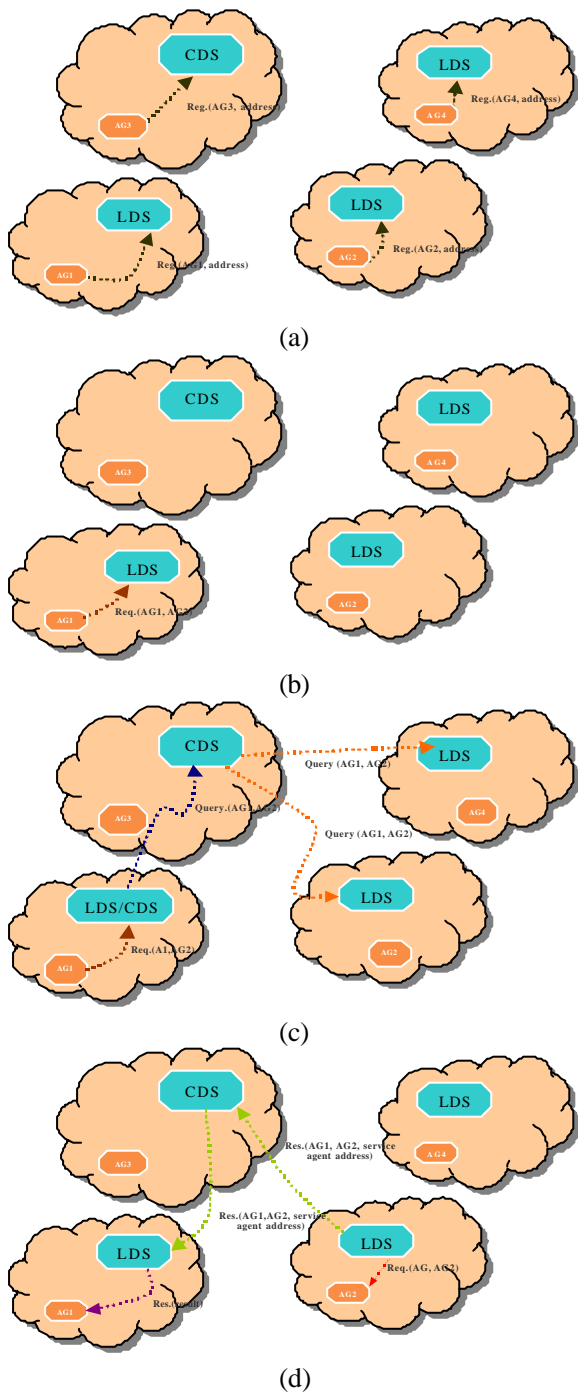


Figure 8: Directory Service queries

4.2 Disadvantages

Of course, every system engineering effort has to consider tradeoffs and potential disadvantages, and the system described in this paper is no exception. A number of issues must be considered before adopting agent technology for SCADA systems. These issues are the following:

- **Network availability**

Even though the SCADA system is integrated and reliable, and thus it can take advantage of the remote execution of their agent tasks in other systems, it is still dependent on the availability of the Internet or other network connection. For instance, if the site cannot connect to the Internet or to another network, the possibility to solve the reliability problem by using

mobile agents is limited to the local system. If the local system is also no longer able to support a network, then the whole architecture becomes ineffective. Therefore, it is important to have a healthy network environment. This however is becoming more and more a necessity in any commercial environment, and it does not impact very much on the site costs.

- **Security**

Whenever a system makes use of a network, potential security hazards arise. They range from simple network downtime problems to more complex intrusions, especially when the system makes use of the Internet as a means of communication between remote locations.

Many solutions have been proposed for this problem. One good “catch-all” solution is to secure the network by means of either a private or a virtual private network system. This will prevent message “snooping” or worse, changing, from unauthorized personnel. At present, TEEMA includes a security communication system based on the Java Secure Sockets Extensions. This system can be further strengthened with the adoption of hardware-based security solutions.

The Java Secure Sockets Extension system also provides a mechanism for secure identification of the communicating partners. This makes it possible to predetermine a set of “trusted” machines that can participate in the control system and exclude intrusions.

Finally, internal agent environment security must be considered. As the system stands now, there is the concrete possibility that an agent crashes an entire machine. To avoid this, it is necessary to implement a safety mechanism for agents, ideally based on the provision of levels of execution that make it possible for a more “tested” agent to run closer to the system, whereas a less “tested” system will not be allowed to use system calls, for example.

- **Dependability**

Although the reliability of the system grows in terms of fault tolerance and fault recovery, it is necessary to consider the fact that a new system is now in series with the SCADA system: the network connection. The reliability analysis must now consider a different scenario (Figure 9).

The trade-off introduced by this new scheme is relatively simple to determine, but it depends on the availability of the network structure and the possibility to make it as reliable as needed. This could also require the use of special network equipment, such as recoverable connection gateways, and has to be included in the economic analysis for the adoption of such a system. Currently, the network availability does not seem to be a problem and most SCADA systems are already using the network to communicate with the main control line.

4.3 Break-even analysis

The careful consideration of these parameters will decide whether it is acceptable or not to adopt the agent architecture in a preexisting infrastructure. For new systems, however, it is necessary to consider that most SCADA systems are built from scratch and thus there is no direct strategic advantage in choosing one over the other. Also, technological changes warrant a rewrite of a SCADA system within two to three years from its initial deployment.

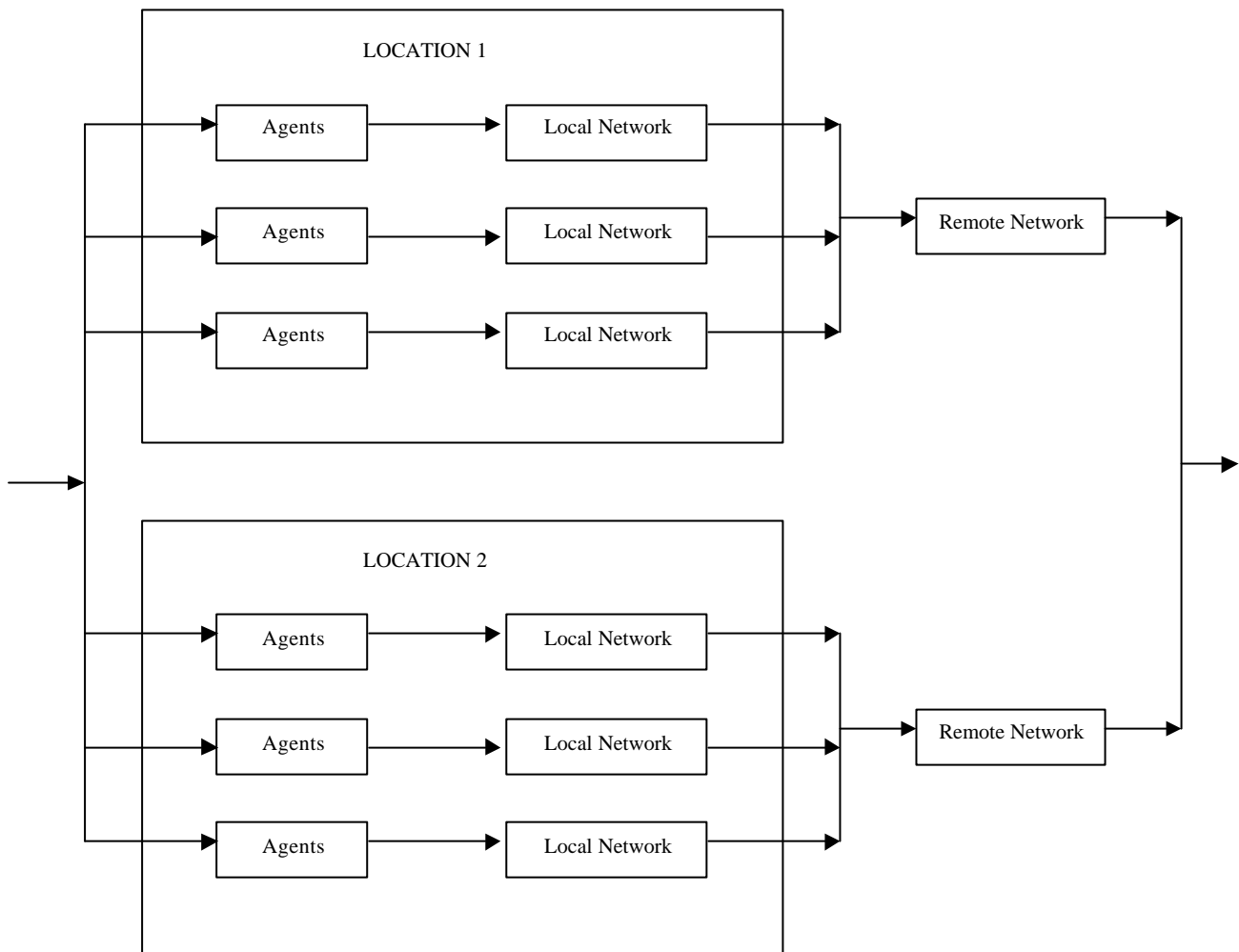


Figure 9: Reliability diagram

5 Conclusions and Future work

In this paper, software agent technology with a directory service has been applied to SCADA system architecture. The directory service is used mainly to provide the subsystem with the capability to locate where other subsystem services are located. This approach is more reliable, versatile, and integrated than the conventional approach. Moreover, the architecture discussed in this paper integrates the network structure in the SCADA system, making it suitable for remote monitoring and control applications.

The system also presents some disadvantages, mainly due to the new distributed architecture. Future work can help overcome these disadvantages. In particular, the following issues appear to be promising:

- **Discovery services**

The inherently distributed structure of the agent environment provides a means for location independence, but the nature of the directory service requires that specific labels be attached to each system. In the current implementation, there must be a central directory system to which every local directory system must register.

Future implementations will implement a discovery service that allows dynamic discovery of directory services with a mechanism similar to ARP address discovery.

- **Over Platform**

Although agents can run virtually on any platform, there is still the necessity to have the same agent execution environment on every system. This can become a difficulty in case agents need to be connected to dedicated environments running specific agent execution environment. This can happen for a variety of reasons, but mostly it is a matter of security: for example, it might be that wireless communications providers trust only their specific agent platform. This can be overcome by automatic translation systems that will allow semantic equivalence of agents in both platforms.

This concept is being elaborated in another form by Web developers with new architectures that become independent from the platform or location. They might be thought of an extension of Object Oriented Programming for Internet applications.

Acknowledgements

The authors wish to gratefully acknowledge NSERC and TRILabs for their partial support of this work.

The authors also wish to acknowledge the support of IRAP through the Technology Transfer Laboratory at the University of Regina.

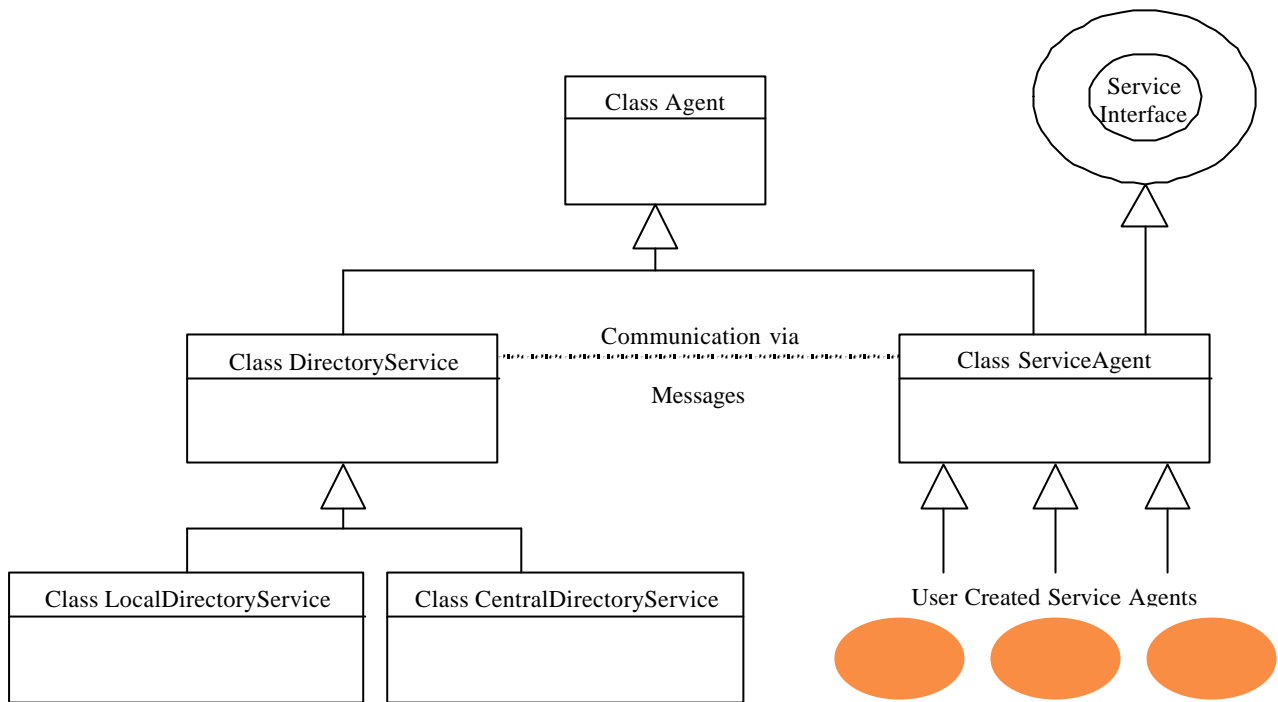


Figure 10: Directory Service class structure

References

- [1] Alfonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna. Understanding Code Mobility. IEEE Transactions on Software Engineering; Volume: 24; No. 5; May 1998. IEEE Press.
- [2] Jin Jing, Abdelsalam Hetal, and Ahmed Elmagarmid. Client-server computing in mobile environments ACM Computing Surveys. Volume: 31; Pages 118-157; Baltimore; Jun. 1999. ACM Press.
- [3] Hyacinth Nwana, and Divine Ndumu. *An Introduction to Agent Technology*. Intelligent Systems Research Applied Research and Technology, BT Labs.
- [4] Traci J Hess, Loren Paul Rees, and Terry R Rakes. *Using autonomous software agents to create next generation of decision support system*. Decision Sciences; Atlanta; Winter 2000.
- [5] Andrzej Bieszczad, Bell Laboratories, Bernard Pagurek and Tony White, Carleton University. *Mobile Agents For Network Management*. IEEE Communications Surveys. Volume: 1; No. 1; Fourth Quarter 1998. IEEE Press. <http://www.comsoc.org/pubs/surveys>
- [6] Mahadev Satyanarayanan. *Accessing information on demand at any location Mobile Information Access*. IEEE Personal Communications; Feb. 1996. IEEE Press.