A FRAMEWORK FOR DISTRIBUTED AND INTELLIGENT PROCESS CONTROL

Qurban A. Memon
UAE University, United Arab Emirates
qurban.memon@uaeu.ac.ae

Keywords: Distributed network control, Process Control, Intelligent agents.

Abstract: The customized development of the Distributed Control System for process control in an environment of intelligent and tagged field devices etc., is the main focus of this work. The proposed solution consists of two-layer approach: use of decentralized intelligent agents at the local process level, and four-tier modular architecture at central controller level to help implement distributed intelligence. The design and development issues for such a customized design are investigated.

1 INTRODUCTION

The decentralization of control, and expanding physical setups have resulted into today's distributed process control (DCS) systems (M. Ioannides, 2004, J. Alonso, 2000). The research in this area is quite active because of these developments, for example, Profibus fieldbus networks and wireless Profibus for real time industrial control systems (E. Tovar, 1999, A. Willing, 2003). Recently focus is reported with respect to distributed intelligence for reduced operational changes. These efforts have respective generated agent based approach (Bernan, 2002, F. Maturna, 2005). Currently, active Radio Frequency Identification (RFID) devices are being deployed for a variety of process control industry solutions (A. Juels, 2003, J. Bohn, 2004). To some industries, RFID is bringing a level of automation and control similar to what process control devices brought to manufacturing decades ago. I. Satoh, 2004, presented a framework which exploits agents to enhance capabilities of the users in an environment of tagged devices. In another work (S. Naby, 2006), the author discusses idea of integrating software agents into RFID architectures to accumulate information from tags and process them for customer/object or system specific use, for example a concurrent mission. As a summary, the optimization in network performance combined with distributed intelligence in an environment of non-stationary and reconfigurable devices provides a new direction of research.

The process under investigation is shown in Figure 1, which shows reprogrammable and reconfigurable control devices including some tagged and distributed field intelligent devices. In short, the challenges for DCS development include process reconfigurability and intelligent decision making within a Profibus/Profinet compatible network. For comparison, a model is to be used to set a baseline for performance. As the network is distributed, hence a multiple input multiple output (MIMO) baseline is considered that requires performance matching to that of the centralized MIMO. For Figure 1 to achieve similar performance to that of the centralized MIMO, each parameter update needs to be communicated over the network at times. In order to categorize time delay $d(t)$ in the network, we divide it into three categories:

$$d(t) = d_1(t) + d_2(t) + d_3(t) \quad (1)$$

where $d_1(t)$, $d_2(t)$, and $d_3(t)$ represent time delay when a device communicates with controller, time delay when a group of devices communicate with controller; and when all devices communicate with controller respectively. One obvious approach could be to minimize either of these delays so as to optimize the performance to match a centralized MIMO system.

2 PROPOSED APPROACH

A set of processes is proposed to introduce intelligence at field level to gain respective

independence. This way, minimized communication with the main controller thwarts communication bottlenecks caused by interoperability of devices, or simple operational requirements at local level. This also improves survivability of the local processes in cases when central controller fails in providing critical timely decision. The configurability of devices may be provided by collecting operational parameters at the device(s) level followed by estimation of parameters of concerned entities at the central level. This leads to two separate domains:

2.1 Local Process

The local entities tend to be distributed throughout the environment to support overall operations. The job of these entities can be done effectively by agents. The agents collaborate, learn and adjust their abilities within the constraints of the global process.

Agent design mechanism: A lot of work is done on agents alone and details can be found in (R. Brennan, 2001) Agents are active software entities that can request for additional capabilities once they discover that the task at hand can not be fulfilled. The programming of these agents is done at the central level where a set of heuristics is used for reasoning at the local level, and is stored as a function block diagram (like an internal script). The agents know about their equipment, continuously monitor its state, and can decide whether to participate in a mission or not. The collaborating agents join (on their own will) and thus form a cluster in order to enable a decision making. In addition to agents, there are other computing units that exist at the local level and help to form a cluster. These are known as cluster directory (CD) and cluster facilitator (CF) respectively. The following steps describe agent collaboration in clusters:

- Agent N receives a request from central process.
- It checks its scripts, and solves local steps.
- For external steps, it receives contact details of other agents with external capability from CD.
- Agent N creates CF_N and passes on these details.
- CF_N passes the request to specified agents, and thus cluster is formed.

For efficient collaboration, CD must remain updated for recording the information of its members, such as agent name, agent locator, service name, service type, and so on. Upon joining or leaving the cluster, an agent must register or cancel registration respectively through CF. Through a query, an agent can find out other members' services and locators. Through these steps, a trust is developed and thus members hold higher authority than non-members.

Recently developed tools may be used to help design cluster facilitator (CF) and domain ontology, using for example DARPA Markup Language (DARPA, DAML, 2000). The DAML extends XML (Extensible Markup Language) and RDF (Resource Description Framework) to include domain ontology. It provides rich set of constructs to create ontology and to markup information for attaining machine readability and understandability. Furthermore, the Foundation for Intelligent Physical Agent (FIPA, 2003) Agent Management Specification is extended to develop the agent role called CF to manage cluster directory (CD) and cluster ontology. Using assistance from DAML-based ontology, the members of the cluster are able to form cluster and communicate with other agents.

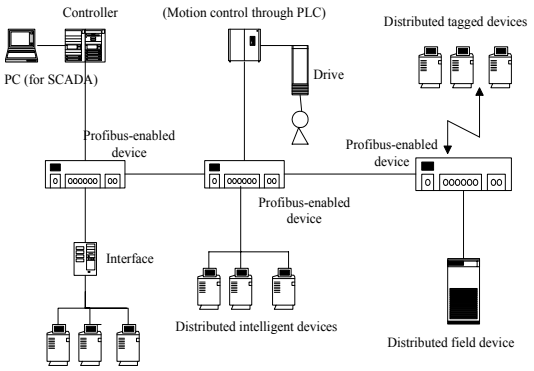

Figure 1: Typical Process Control Network.

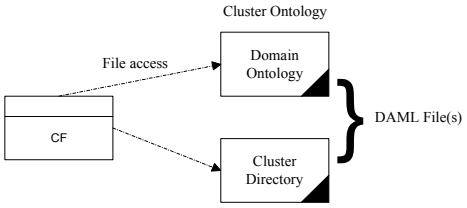

Figure 2: Linking CF with DAML.

The interaction among domain ontology, CD and CF can be best understood using Figure 2. Figure 2 shows how CF gets access to DAML files and facilitates the common goal of the cluster. There are tools available like Jena semantic web that can be used to handle the cluster director (CD) built using DAML, and to develop a Java class "Directory". Thus, main functions of CD can be summarized, as:

- Add and Remove the information of an agent
- Get the list of agent names of all members
- Get the information of individual agent by name

- Get ontology used by members in the cluster
- Add external ontology if provided by an agent

Using local process mechanism and main functions of CD, the partial directory can be described as shown in Figure 3. It shows information of CF (lines 1-9) and members of cluster (lines 20-22), the cluster directory also records meta-data about cluster such as cluster name (line 12), cluster description (lines 13-15), ontology used in cluster (lines 16-18), etc.

An example can be illustrated to show how ontology may be updated (Fig. 4(b)) and that how interactions may develop in a local process. It should be noted here that basic cluster ontology provided by CF remains the same but all members' domain knowledge (ontology) may not be the same. For example, user agent holds basic knowledge of the local process but does not understand the knowledge that a distributed field device holds. Through DAML-based ontology, members can communicate with each other to acquire requested service, as shown in Figure 4. It is clear from Figure 4 that when distributed field device agent joins the cluster, it informs CF about corresponding ontology it provides (Figure 4(a)). Thus the CF maintains local process ontology plus the distributed field device ontology. When a user agent wants to perform a task, it asks CF about domain ontology and the agents that provide external capability. In response, CF informs the user agent if ontology is to be acquired (Figure 4(c)). Thus, the user agent can communicate with the distributed field device agent (Figure 4(d)).

2.2 Central Process

This process embodies core, like definition of controller tasks, and definition of domain ontology of each cluster. The other components are removal of agent deadlocks, estimation of local characteristics and decision making in cases when situation develops beyond the capabilities of agent clusters. It can be argued that if only small scale changes are to be decided at the central level like reconfiguration of device processes then intelligence can further be distributed to the agents at local level. In Figure 5, the model architecture of four tiers is shown to implement objectives of the central system. At the bottom layer (Tier 1), active readers or Profibus/Profinet enabled devices collect data, often collected on a trigger similar to a motion sensor. These readers should be controlled by one and only one edge server to avoid problems related to network partitioning. This layer also provides hardware abstraction for various Profibus/Profinet compatible hardware and network drivers for interoperability of devices. The edge sever (Tier 2) regularly poll the readers for any update from device agents, monitors tagged devices and distributed devices through readers, performs device management, and updates integration layer. This layer may also work with system through controls and open source frameworks that provide abstraction and design layer. The integration layer (Tier 3) provides design and engineering of various objects needed for central controller as well as for field processes and for simulation levels of reconfigurability. This layer is close to business application layer (Tier 4). The monitoring of agents behavior, its parameters and cluster characteristics are done at this layer to assess the degree of reconfigurability.

```
1. <cluster:CF rdf:ID="theCF">
2. <cluster:agentName>"CF"</cluster:agentName>
3. <cluster:agentDescription>
4. "DCS Cluster Facilitator"
5. </cluster:agentDescription>
6. <cluster:locator>
7. "http://dcs.ee.uaeu.ac.ae/DCS/agent/CF"
8. </cluster:locator>
9. </cluster:CF>
10.
11. <cluster:Cluster rdf:ID="DCSCluster">
12.<cluster:clusterName>"DCS"</cluster:clusterName>
13. <cluster:clusterDescription>
14. "Distributed Control System"
15. </cluster:clusterDescription>
16. <cluster ontology>
17. "http://dcs.ee.uaeu.ac.ae/DCS/ontology/dcs.daml"
18. </cluster:ontology>
19.
20. <cluster:hasCF rdf:Resource="#theCF"/>
21. <cluster:consistOf rdf:Resource="#agent1"/>
22. <cluster:consistOf rdf:Resource="#agent2"/>
23. </cluster:Cluster>
```

Figure 3: DCS Cluster Directory.

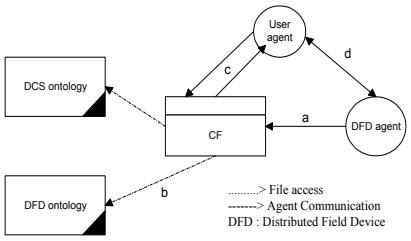

Figure 4: Ontology update provided by DFD.

This layer also takes care of parameters like handling device processes, resource allocation and scheduling of processes. The separation of edge server and integration layer improves scalability and reduces cost for operational management, as the edge is lighter and less expensive. The processing at the edge reduces data traffic to central point. Similarly, the separation of integration from business applications helps in abstraction of process entities.

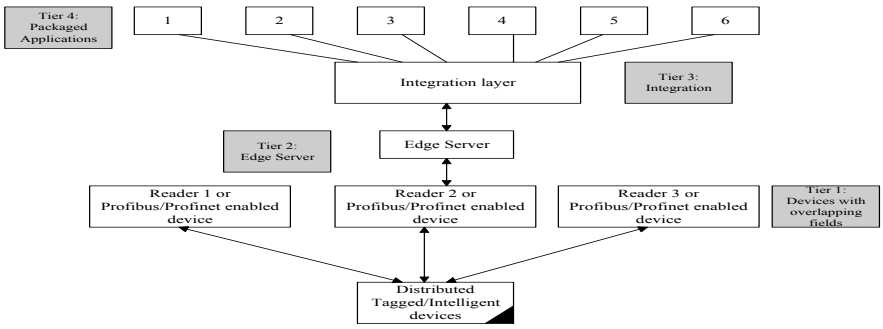

Figure 5: 4-Tier Reference Architecture.

The Tier 3 also enables it as self-healing and self-provisioning service architecture to increase availability and reduce support cost. Control messages flow into the system through business application portal to the integration layer, then on to the edge and, eventually, to the reader. Provisioning and configuration is done down this chain, while reader data is filtered and propagated up the chain.

3 CONCLUSIONS

The main idea behind two processes is decentralization. The communication delay is reduced at the cost of increased intelligence at the local level. In fact, if we look at equation (1) we see that $d_1(t)$, $d_2(t)$ and $d_3(t)$ minimize to a level when problem of the node device exceeds the threshold level of the agent intelligence. If collaborative intelligence exceeds combinatorial complexity then there is no need of communication between devices and the controller and requirements of the central process reduce to that of the design of agents only. Thus, the performance matches to that of the centralized MIMO system. The four-tier modular architecture at central level helps in implementation of distributed intelligence at field level and in designing of agents. The functionality more appropriate to the layer has been fit into respective tiers at central level. Additionally, design and reconfigurability can help introduce features in agents to thwart intrusive agents, during real time. This set of gains has not been claimed in either of the approaches (E. Tovar, 1999, A. Willing, 2003).

ACKNOWLEDGEMENTS

This work was financially supported by UAE University under a grant no. 01-04-7-11/07.

REFERENCES

A. Juels, R. Rivest, M. Szydlo, “The Blocker Tag: Selective Blocking of RFID Tags for Consumer Privacy”, *Proceedings of ACM conference on Computer and Communications Security*, 2003.

A. Willig, “Polling based MAC protocols for improving real-time performance in a wireless Profibus”, *IEEE Trans. on Industrial Electronics*, pp. 806-817, 2003.

Bernnan, M. Fletcher, and D. H. Norrie, “An agent-based approach to reconfiguration of the real-time distributed control systems,” *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 4, 2002.

DARPA Agent Markup Language, *http://www.daml.org/*

E. Tovar, and F. Francisco, “Real-time field bus comm. using Profibus Networks”, *IEEE Trans. on Industrial Electronics*, pp. 1241-1251, 1999.

F. Maturana, R. Staron, K. Hall, “Methodologies and Tools for Intelligent Agents in Distributed Control”, *IEEE Intelligent Systems*, pp. 42-49, February 2005.

Foundation for Intelligent Physical Agents (FIPA), *http://www.fipa.org/specs/fipa00023/*

I. Satoh, “Software agents for ambient intelligence”, *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pp.1147-1150, 2004.

J. Alonso, et al, “Development of a Distributive Control Scheme for Fluorescent Lighting based on LonWorks Technology”, *IEEE Transactions on Industrial Electronic*, Vol. 47, No. 6, pp. 1253-1262, 2000.

J. Bohn, F. Mattern, “Super-Distributed RFID Infrastructures”, *Lecture Notes in Computer Science (LNCS)* No. 3295, Springer-Verlag, pp. 1-12, Eindhoven, Netherlands, November 8-10, 2004.

M. G. Ioannides, “Design and Implementation of PLC based Monitoring Control System for Induction Motor”, *IEEE Transactions on Energy Conversion*, pp. 469-476, 2004.

R. Brennan and D. Norrie, “Agents, holons and function blocks: Distributed intelligent control in manufacturing”, *Journal of App. Sys Studies*, 2001.

S. Naby, and P. Giorgini, “Locating Agents in RFID Architectures”, Technical Report # DIT-06-095, University of Trento, Italy, December 2006.