On Interoperability and Intelligent Software Agents for Field Area Networks

Peter Palensky

Institute of Computer Technology
Vienna University of Technology
Gusshausstr. 27-29/384
A-1040 Vienna, Austria

Abstract. Intelligent software agents are software programs that are currently used to solve complex problems in the domains of network management or information retrieval. The usage of agent technology for field area network would offer a lot of new and enhanced applications and would be “the second chapter” of distributed applications. The missing link to the commercial exploitation of agent technology on field area networks is a standard for agent communication. This paper gives an overview and takes the first steps towards standard agent communication for field area networks.

1 Special properties of field area networks

The large variety of field area networks (FANs) makes a general description almost impossible. But there are still some basic properties that are common to most of them.

The first one is that FANs usually communicate via a serial communication channel. This channel can be for instance a twisted pair wire, a radio channel or a fiber optical cable. Based on this physical channel the members of the fan have to arbitrate and share this common communication means. Only one of the FAN nodes is allowed to talk, while the other may only listen. There is of course the possibility of setting up separate segments, connected via routers or bridges, but this problem then still exists within one segment or via one router. This communication channel is usually relatively slow. The low bandwidth of the FAN communication channels results in an increased robustness, which is more important for FAN applications than a high throughput [2].

Another property is that FANs tend to keep computation in the field. Modern field area networks consist of a number of small nodes that usually incorporate various means for sensor/actor interaction with the environment and a microprocessor. This microprocessor is able to execute parts of a distributed application. A temperature sensor node for instance operates the sensor and its analogue digital converter (ADC). The other nodes that need the sensed value do not have to care about the timing of the ADC or the linearization or calibration of the sensor. They get the sensed value without caring about if the ADC is 8 bit or 12 bit wide, what is its offset and so on. The other nodes receive the temperature value in degrees Fahrenheit or Celsius already formatted in the right and understandable way. Most of these microprocessors are 8 bit (or less often 16 bit) controllers with very low computational power and memory.

The microprocessor has usually two main tasks. The first one is the handling of the FAN communication protocol. Sometimes the lower layers are handled by hardware circuits but the usually most of the layers are implemented in software. The second task is the application itself. These applications can reach from simple light switch program to more complicated thermostat controllers. The programming languages for these controllers are either assembler or some dialects of high level programming languages like ANSI-C or Pascal [4, 3]. These high level programming languages for FANs are designed especially for real time applications and hardware oriented programming.

The last common property for FANs that will be mentioned here is their message oriented communication. Measuring values or commands are communicated via short messages. FANs are usually not designed for continuos data streams like needed for video cameras or audio transmission. Other types of networks are more suitable for these kinds of application [1].

2 Interoperability for Field Area Networks

Communication of field area networks is based on communication protocols. These protocols are usually using a layered architecture. The lower layers are more concerned with the physical properties of the communication system, while the upper layers tend to be more abstract. The ISO-OSI seven layer model is a well known example for a layered protocol suite and is used as reference for all kinds of layered protocols [14]

The communication protocols describe the rules how to transmit data. The structure of packets and frames, how to establish a connection and other aspects of data transmission are defined in the protocol specifications. But these specifications are not sufficient for real *understanding*. Two network partners that want to exchange information need to settle on certain rules that exceed ordinary protocol conformance [2]. Interoperability guidelines define how to encode and format physical values that have to be transmitted over the network. They define the syntax and the semantics of communication objects and provide functional profiles for the components of distributed applications.

Without these interoperability guidelines a packet could be transmitted in a proper way (means the bits are in the right order etc.), but the recipient would not know if that packet represents a temperature value or a command to turn on the light. The interoperability guidelines define the structure of the application layer interface of the field area network nodes. With these guidelines it is possible to identify, distinguish and assign transmitted data and its functionality.

Various types of field area networks have interoperability guidelines for HVAC-systems, lighting, sunblinds or fire alarms. These profiles are to be configured and installed by a network management tool that knows the their purpose and functionality.

LonMark, the interoperability association for LonWorks, for instance defines standard network variable types (SNVTs) and functional profiles (LonMark profiles) [11, 10]. EIBA, the EIB association, defines EIB interworking standards (EIS) and object interworking standards (ObIS) for the European Installation Bus (EIB) [12]. Both interoperability guidelines define types for physical values like temperature, speed or humidity. The profiles describe how to use this types to create an interoperable application layer interface. This paper uses LonWorks and EIB as examples for FANs.

3 Intelligent Software Agents

Intelligent Software-Agents (or short agents) are autonomous and usually distributed software programs that communicate and cooperate with other agents and humans. They try to reach a given goal without any further interaction with their creator (another agent or a human) by using sensors, actuators and communication. Agents are widely used in the Internet for information retrieval and filtering. [9] describes the properties of agent and its differences to ordinary software programs as being autonomous, goal-oriented, learning, flexible and others.

One of the most important abilities of agents is to cooperate. Usually cooperation is based on communication, and therefore agents have to take part in a kind of communication network. Agents use agent communication languages (ACLs) to exchange data, queries and other kinds of information via a network. Well known examples for ACLs are the knowledge query and manipulation language (KQML) [5, 6] and the FIPA (foundation for intelligent physical agents) ACL [8]. These ACLs are implemented for various platforms and networks like the world of TCP/IP and Java. This paper focuses on KQML which offers the mechanisms for establishing and routing messages without the need of a specified language of the actual content. It is just more “wrapping” protocol than a protocol.

4 Intelligent Agents on Field Area Networks

The convergence of agent technology and field area networks can be done in two ways. The first approach is that agents are just used as a metaphor for component- or object oriented development.

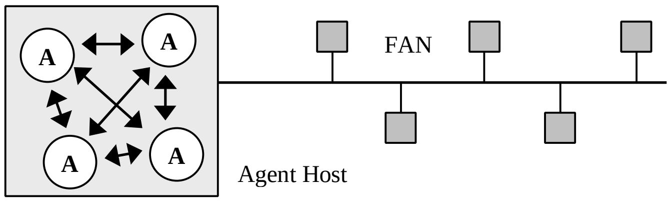

Fig. 1. Agents on an agent host

An application is split up into individual processes that are executed on an agent-host like depicted in Fig. 1. The FAN is simply the sensor- and actuator interface for the agent society, the inter-agent communication is done on the agent-host. The advantage compared to a monolithic program is its structure, the agents can be simple and robust program components that are more easy to program and maintain.

The second way, and that is the one that is investigated here, results in distributing the agents in the FAN (Fig 2.). They are executed on the FAN nodes and communicate via the FAN, therefore the ACL has to be mapped onto the FAN protocol.

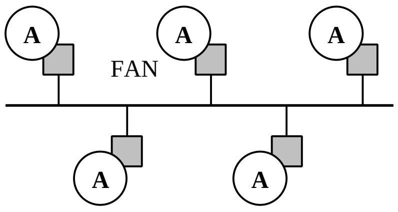

Fig. 2. Agents on FAN nodes

This creates a couple of problems. FAN-nodes are probably to weak for agents and the bandwidth is not suitable to transfer large messages, databases or knowledebases.

Special types of agents are required. They have to be compact, robust and simple. They have to get along with a minimum of communication, preferably based on messages. So called “reactive” agents meet these requirements [15]. They can not be called intelligent on their own, but a multi agent system (MAS) built up with reactive agent shows intelligent behavior [16]. With these agents it is possible to realize efficient problem solvers and complex algorithms although only primitive nodes are used [17]. The functionality of the network is modeled with agent structures, that can be developed and simulated on workstations using suitable simulation environments.

5 FAN-Interoperability for Agents

We suppose a field area network that connects agent-hosting nodes. What is missing is a data-type for agent communication and a functional profile for agents.

Two basic steps have to be done for agent interoperability on FANs. First a SNVT or EIS-type for KQML-messages has to be standardized. Second a profile for general purpose-agents has to be defined. After this syntactical agreement the content and the content-language of the KQML-message is still undefined.

A KQML message has the following format [7]:

```
(<speech act type>
     :content <statement/speech act type>
     :sender <name>
     :receiver <name>
     :language <text>
     :ontology <text>)
```

One proposal could be to define a FAN-type for every speech act type. The other possibility is to define one FAN type for all KQML messages and to distinguish the different speech acts inside the type. In the case of LonMark SNVTs the latter would be the right choice, in case of EIBA EIS the first one. EIS9 for instance is suitable to contain various physical values like temperatures, voltages or speeds. The same can be done for the KQML speech acts.

The sender and receiver address section of the KQML message can be abandoned when it is ported to a FAN packet, because this information is then already included. There is no existing standard for the content language even for existing Internet agents. There are approaches to use KIF (knowledge interchanging format), but this ASCII-encoded language would enlarge our packets too much. A more FAN-specialized approach is necessary where every bit of a message is used. A reasonable approach would be to map KIF symbols on numerical tokens. The ontology is domain and application-specific. The language and ontology fields in the KQML message can be sufficiently covered by a couple of bits.

So it is possible to port KQML messages to FAN variable types. These types can then be used for functional profiles. The application layer profile for an agent would look quite simple (Fig. 3).

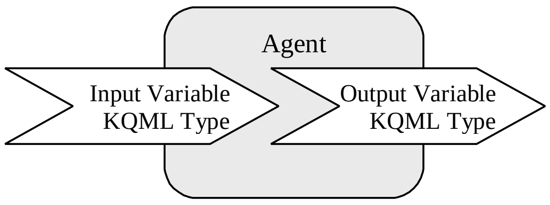

Fig. 3. The functional profile of an agent

One input and one output variable cover the needs for agent communication. This profile should only be used for agent specific aspects of the distributed application like knowledge exchange, cooperation and learning. Ordinary values or commands should not be transferred via this KQML interface but use the ordinary variables. Otherwise it would be a simple protocol tunneling and all nodes would just consist of these two variables.

So an agent enhanced FAN node is an ordinary node with a couple of communication objects in his profile, including one agent object.

6 Conclusion

This paper showed that intelligent agents can be used for FAN applications. They help to make a complex and distributed system more structured and more powerful. It is possible to implement and use them now, without any standards and agreements on communication means. The goal should be to find a standard which can be used for future applications for interworking and cross vendor communication.

References

1. IEEE Standard for a High Performance Serial Bus, IEEE Std. 1294-1995, ISBN 1-55937-583-3, IEEE Computer Society, 1995

2. D. Dietrich, D. Loy and H.-J. Schweinzer: “LON-Technologie”, Hüthig Verlag, Heidelberg, Germany, 1998

3. “Process Pascal Users Manual”, Version 2.0, Proces-Data A/S, Silkeborg, Denmark, 1991

4. “ Neuron-C Programmers Guide”, Revision 4, Echelon Corporation, Palo Alto, USA, 1995

5. Tim Finin, Yannis Labrou and James Mayfield: “Software Agents, chapter KQML as an agent communication language”, MIT Press, 1997

6. Yannis Labrou and Tim Finin: “A semantics approach for KQML -- a general purpose communication language for software agents”, Third International Conference on Information and Knowledge Management (CIKM'94), 1994

7. James Mayfield, Yannis Labrou, and Tim Finin: “Evaluation of KQML as an Agent Communication Language”, in Intelligent Agents Volume II - Proceedings of the 1995 Workshop on Agent Theories, Architectures, and Languages. M. Wooldridge, J. P. Müller and M. Tambe (eds). Lecture Notes in Artificial Intelligence, Springer-Verlag, 1996

8. Donald Steiner: “Die FIPA-Initiative für Agenten-Standardisierung”, in: “it+ti”, Vol. 4/98, ISSN 0944-2774, Munich 1998

9. S. Franklin and A. Graesser: “Is it an agent, or just a program? A taxonomy for autonomous agents”, in Jörg P. Müller, Michael J. Wooldridge and Nicholas R. Jennings (editors): “Lecture Notes in Artificial Intelligence 1193, Intelligent Agents III, Proceeding of ECAI’96 Workshop”, ISBN 3-540-62507-0, Springer, 1997

10. LonMark Application Layer Interoperability Guidelines, Version 3, LonMark Interoperability Association, USA, 1996

11. LonMark Layer 1-6 Interoperability Guidelines, Version 1.3, LonMark Interoperability Association, USA, 1994

12. EIBA Handbook for Development, Issue EIB 2.21, 12/95

13. O. Etzioni, S. Hanks, T. Jiang, R. M. Karp, O. Madani and O. Waarts: “Efficient Information Planning on the Internet”, Proceedings FOCS-96, 1996

14. A. S. Tanenbaum: “Computer Networks”, Second Edition, Prentice Hall Inc., 1998

15. Darryl N. Davis: “Reactive and Motivational Agents: Towards a Collective Minder”, in Jörg P. Müller, Michael J. Wooldridge and Nicholas R. Jennings (editors): “Lecture Notes in Artificial Intelligence 1193, Intelligent Agents III, Proceeding of ECAI’96 Workshop”, ISBN 3-540-62507-0, Springer, 1997

16. R. A. Brooks: “Intelligence without representation”, in Artificial Intelligence Vol. 47, pp139-159, 1991

17. P. Palensky and M. Gordeev: "Demand Side Management by using distributed artificial intelligence and fieldbus technology", IARB99 Intelligent and Responsive Buildings Conference, Brugge, Belgium, 1999