Journal of Computer Science 2 (1): 92-96, 2006
ISSN 1549-3636

Managing a High Speed LAN using Distributed Artificial Intelligence

Ibrahiem M.M-El-Emary
Faculty of Engineering, Amman Al Ahliyya University, Amman, Jordan

Abstract: This study is concerned with a practical application of distributed artificial intelligence for managing the high data rate bus structured local area computer network that uses deterministic multiple access protocol. In the selected network that is managed using distributed artificial intelligence, the dynamic sharing of the available bandwidth among stations is achieved by forming "train to which each station may append a packet after issuing a reservation. Reservation and packet transmissions are governed by the reception of control packets (token) issued by the network end stations. Managing approach that was suggested depends on using intelligent autonomous agents, which are responsible for various tasks among it: election of the end stations, the recovery from failures, and the insertion of new stations in the network. All these tasks are based on the use of special tokens.

Key words: Distributed artificial intelligence, autonomous agents, neural networks, EXPRES-NET, CSMA/CD

INTRODUCTION

The main task of the network management system researchers is to develop new tool that work better than current by available tools for doing this laborious work. The work presented in this study directed about how to delegate as much as possible to the machine, using network administrators as knowledge engineers that teach the machine how it should perform its work based on what is called Intelligent Autonomous Agents.

There are a few predictable advantages of using Intelligent Autonomous Agents for any management system, network management being just one. First of all, there is the intelligent nature, which sounds promising. Having an intelligent and adaptable system is usually better than having dedicated applications for specific solution. Likewise, the word Autonomous gives us the idea of something that can work by itself or need almost no human interference. Finally, agent gives the impression of a helper or a wizard that some how works between the machine and human. So, the main objective proposed here is to project a system where the human system administration does not have to work as the truly intelligent-meaning cognitive- element in the management process, feeding the system with the rules of work or knowledge-it needs to operate, no more boring work, but intelligent work[1].

The network that is managed by the above approach categorized as a local area network labeled by TOKENET[2], its architecture based on a linear bidirectional bus, and on a deterministic channel access protocol, whose efficiency is similar to that of EXPRESS protocols[3]. Advantages of this network over the linear topology (L-EXPRESS-NET) are that stations need not be numbered, and that no silence counter is needed in order to schedule transmissions. The used LAN is named TOKENET since tokens (control packets) are used in the operations of the channel access scheme. The protocol algorithms were developed for a scenario in which Ethernet-like networks cannot be used, i.e. considering a large population of users that can be connected using several kilometers of cable, and whose traffic requirements must be satisfied with a very high data rate (>100 Mbps). The packet transmission time is assumed to be much shorter than the end-to-end propagation delay, so that the CSMA/CD yields extremely poor performance.

Architecture of the TOKNET Network: TOKENT is based on a linear topology. It is thus possible to identify the rightmost and leftmost active stations (where by active we mean that stations are executing the protocol algorithm). These two stations are named "end stations". Stations are connected to the bus by means of passive taps that can simultaneously receive and transmit[4]. Packet collisions can be detected. Interfaces are capable of recognizing a limited set of short control packets, named tokens, whose functions will be described later on. The reception of tokens as well as the reception of packets may produce some response by a station. The time needed in order to start this response is $\mathbf{t_d}$ seconds; this includes the time for the recognition of the token (or the detection of the end of a packet reception) and the time to initiate the response. $\mathbf{t_d}$ should be of the order of a few bit times[2].

Data packets are transmitted in trains, composed of a token (the locomotive) followed by some packets (possibly none, in which case the train is said to be empty). The time separation between successive elements of the same train is $\mathbf{t_d}$ seconds. Packets may

Corresponding Author: Ibrahiem M.M-El-Emary, Faculty of Engineering, Amman Al Ahliyya University, Amman, Jordan

board the train only after a reservation has been issued. By listening to other stations reservations each station can determine the correct position of packet in the train.

Transceivers can operate in four different modes depending on the state of the station: the CSMA-CD mode and the polite sensing mode are used during system initialization in order to elect the end stations; the reservation mode is used when a station has a packet ready for transmission in order to reserve a place on the next train; the scheduling mode is used after a reservation has been made in order to transmit the packet in the train in the reserved position.

The CSMA-CD mode implies the transmission of packets following the CSMA-CD protocol[4,6]. It is assumed that $\mathbf{t_c}$ seconds are necessary to recognize a collision. The time $\mathbf{t_c}$ is a few times longer than $\mathbf{t_d}$. When a station is in the polite sensing mode it starts transmitting its packet after recognizing the end of a packet transmission from another station. If a collision is detected in the initial $\mathbf{t_c}$ seconds the transmission is aborted and a new attempt will be made at the end of the colliding packet. When a station is in the reservation mode it transmits a short burst (any predefined bit pattern) when it receives the reservation token. Stations in the scheduling mode count other station reservations, determine which their received wagon in the train is and transmit their own packet in the reserved position. If a collision occurs in the initial $\mathbf{t_c}$ seconds the transmission is immediately halted**.** Receivers are always enabled. Packets (and burst) must be separated by at least $\mathbf{t_d}$ seconds to be received.

Packets can be of variable length. Before each packet a preamble is transmitted in order to synchronize the destination station receiver. The preamble duration is of the order of several tens of bits. In some cases a long preamble is necessary. It is not necessary, insteade, to transmit a preamble before the reservation bursts, since they must only be detected and need not be decoded.

Access scheme used in TOKENET network: Assume the two end stations have been found, and denote, as in Fig. 1, by L the leftmost active station, and by **R** the rightmost active station. Upon recognizing the end of a train**, R** issues an R-token, denoted by $\mathbf{T_r}$, and after $\mathbf{t_d}$ seconds transmit a reservation burst. The token and the burst propagate towards all stations to the left of **R.** Each station with packets ready for transmission, upon reception of $\mathbf{T_r}$ immediately transmits a burst of short duration that serves as a packet reservation. Due to the station latency time the burst is separated from $\mathbf{T_r}$ by a gap of duration $\mathbf{t_d}$. Denote by $\mathbf{t_b}$ the time duration of the burst. Provided that the distance between any two adjustment stations is larger than the distance over which the line signal propagates in $\mathbf{(t_d+t_b)/2}$ seconds, stations receive bursts transmitted from station to the left, separated by intervals of duration larger than $\mathbf{t_d}$**.** It is thus possible for each station to count the number of bursts transmitted by stations to the left. Denote by $\mathbf{N_j}$ the count of station j. Bursts transmitted by the station to the right are instead seen as completely overlapping and are thus undistinguishable.

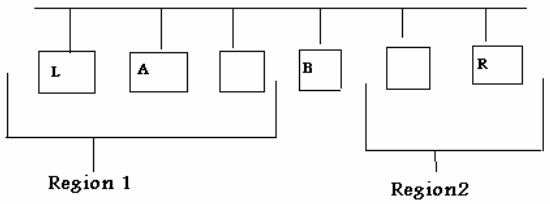

Fig. 1: TOKENET topology; L and R are the network end stations

Finally $\mathbf{T_r}$ reaches **L**, that may transmit a reservation burst as any other station if it has a packet ready for transmit ion. After either transmitting the burst or just hearing the overlapped bursts of all stations with ready packets**, L** issues on **L-**token **T1** that serves as locomotive of a train of packets. If **L** has just transmitted a reservation burst it also append the data packet to **T1**, leaving a silent gap of duration $\mathbf{T_d}$ in between. The train propagates now in the **L** to **R** direction and those stations that just made a reservation may append packets. Before appending its own packet to the train station j must recognize **T1**, and count $\mathbf{N_j}$ packets in the train beyond **T1.** The train propagates all the way to **R** collecting new wagons. **R** may append a packet to the train too, and when **R** recognizes the end of the train it issues a new **R** token, thus starting a new cycle.

A fault in the operation of the station may cause a truncation of the train: if a station has a wrong reservation count, it will either transmit too early (the transmission is stopped after $\mathbf{T_c}$ seconds because of the collision), or not transmit because it will not find enough packets in the train before its own. Both events prevent all stations to the right of the faulty one to transmit in the current train. The same thing happens if the station issues a reservation and then dose not transmit the packet. If instead, a station transmits a packet without transmitting the corresponding reservation, a collision may take place id some station to the right transmits a packet, but other stations can still board the train. Otherwise, if no collision occurs, then the train maybe larger than expected. All above faults conditions only influence next cycle. Other faults concerning either token losses or failures of the end station must be reserved through a system reinitializing.

Quantatively, when we show the impact of the constraint on the minimum distance between adjustment stations, we see that for a **100 Mbps** transmission speed, assuming that the sum $\mathbf{(t_d+t_b)}$ is equivalent to ten bit times, and that the propagation delay is **10μs/km**. The time interval corresponding to

five bits is 50 ns, that is equivalent to a five-meter distance. Stations must hence be separated by at least five meters of cable.

Initialization procedure for TOKENET using CSMA/CD protocol: If an active station doesn't hear any packet, token, or reservation burst on the bus for more than T_o seconds (where T_o is of the order of several round trip delays), it goes onto initialization mode in order to elect the end station and restart the protocol operation. The initialization procedure consists of the following steps:

(a) Using the **CSMA/CD** protocol one station (say station **A**) acquires the channel and issues an initialization token. (Note: this may require several retransmission attempts. At each attempts only the station, which collided in the previous attempt, participates). To avoid ambiguities the initialization token length is such that transition time is longer than the end-to-end propagation delay.

(b) Upon hearing the initialization token, the remaining active stations compete again (as in step 1) until one station (say station **B**) acquires the channel and issues a response token.

(c) Station **A** issues a token which is called token **A**. Station **B** issues token **B** immediately upon hearing token **A.** All the other stations are listening and can determine their relative position with respect to station **B** as follow: for the station in region **1** (Fig. 1) there is a gap (larger than T_d seconds) between the time token **A** is received and the time token **B** is received. For the stations in region **2** token **A** is received immediately (more precisely t_d seconds) before token **B**. Station in region **1** are temporarily inhabited to respond to **R** station location tokens (see next step).

(d) Station **A** issues an **R** station location token that starts a train of tokens since all stations in region **2** (stations that are not inhabited) append a similar token to the train using the polite sensing mode. Each station monitors the channel after transmitting its token. If a station does not hear any other transition after its own it concludes that it is the right end station **(R)**. The token has a special format, namely, the preamble length is such that reamable transmission time is larger than the end-to-end propagation delay. This is necessary to guarantee that all region **2** stations correctly receive tokens.

(e) The newly elected **R** station issues an **L** station location token, and the left end station **(L)** are located using the same procedure above. Again the token preamble transmit ion time is longer than the end-to-end propagation delay.

(f) The **L** station issues an **L** token to notify the end of the initialization phase.

(g) The R station receives the L token and issues an **R** token, thus starting the data transfer phase.

Operation concept of failure recovery and inserting a new station: The failure of an intermediate station (i.e., a station located between **L** and **R**) has no impact on the protocol; thus it does not require a specific recovery procedure. The failure of left or right end station, instead, requires the selection of replacement. Assume that the **R(L)** station fails. The **L(R)** station will detect the failure from the fact that the **R**-token (**L** token) does not come back after the line has been idle for a round trip time. After learning that the **R (L)** station is down, the **L(R)** station decides it is the new **R** station, issues an **L** station location token, and the initialization procedures is executed step 5 on. At the end the new end stations are elected. Note that if the **L** station fails, the old **R** station remains the new **R** station after the recovery, whereas, if the **R** station fails, the old **L** stations becomes the new **R** station.

When a new station become active, it is necessary to allow it to join the other stations in the dynamic sharing of the communication channel. If the new station is in an intermediate position, i.e, between station **L** and station **R**, then it may immediately participate in the protocol procedures. If, instead, the station is either off to the right or off to the left, then it is necessary to reinitialize the whole network. The decision the new stations must take whether to start the initialization procedure itself by listening to the channel.

If the new station receives the **R** token and reservation bursts immediately followed (after t_d seconds) by the **L** token, then it concludes it is off to the left. If the new station detects the end of the train immediately followed by an **R** token then it concludes that it is off to the right. If the new station, by listening to the channel, determines that it is not located between **L** and **R**, then it decides it is the new **R** station, and issues an **L** station location token, thus triggering the execution of the initialization procedure from step 5 on. At the end of the execution of the algorithms the new end stations are elected, and the new station has become station **R**.

TOKENET network management using autonomous agent approach: The main objective in this session is to show how the agents fit in a standard network management system. The used approach depends on presenting a diagram from a regular system and then replaces its internal component by another diagram; using agents based components with similar functionalities.

An ordinary network management could have a workflow like the one presented in Fig. 2[5] in which data is collected from another managed device (1) using some standard management protocol as SNMP. After that, in step (2), the collected data is analyzed, and then

condensed. In step (3) creating the real management information. Later on, managers make their analysis in step (4) and, if required, take reaction in order to correct weak points of the system.

Based on Fig. 2, we have replaced the management approach with another one shown in Fig. 3 adapted to agents' technology oriented. The types of used agents are: data collected (AgF) used to coordination of knowledge exchange between agents of the same community.

In the following, we describe each part of the generic agent's framework shown in Fig. 3 as follows:

(a) Data collector agents (AgP); this type of agents retrieve data from devices and saves this information on a local knowledge database. This knowledge could be kept and processed locally and then shared to a whole community through the agents' framework shown in Fig. 4. The most part of data collector agent functionality-which is collecting data- is implemented by special knowledge rules loaded inside its knowledge database. These rules, in association with native knowledge implemented in the code of the agent, allow them to interface managed equipment through SNMP protocol. A signal data collector may collect information from several different devices. It could be setup with multiple different goals, with one or more goals for each device.

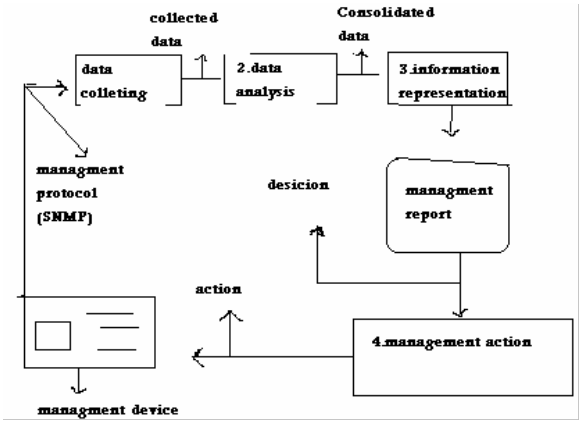

Fig. 2: Network management workflow

(b) Management agent (AgM); are responsible for the organization of operation between agents of the same community .In addition, they can implement a higher level of data analysis based on shared knowledge (data) coming from several different collector agents. Finally, management agents coordinate multiple data collector agents on working on a special set of devices and later correlating that data. Besides, we have other types of management agents associated to agent community and not directly to network

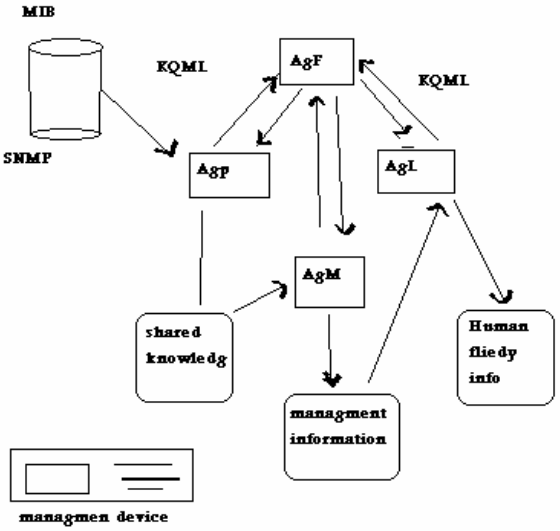

Fig. 3: Network management with agents

Symbols used in Fig. 3 are given by the following:
AgP: collector agent
AgM: manager agent
AgI: Interface agent
AgF: Facilitator agent
KQML: knowledge query management language

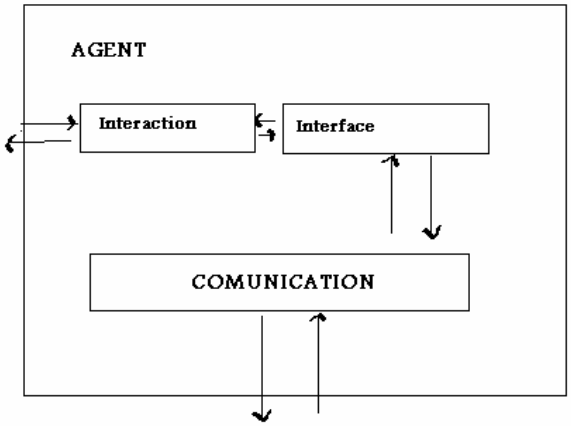

Fig. 4: Generic agent structure

management environments. What we mean by this is that those agents will be presented on any agent community, not just on those dedicated to management self-creation, agents that neural network structure, wizard agents-those that hold and share massive knowledge data base-and many others being projected in order to create an agent virtual world.

The types of agents specially setup for network management purposes are:

* **Data analyzer;** which analyzes data retrieved from different knowledge bases, usually from distinct data collector agents.

* **Data consolidator;** which combines or correlate data coming for a set of distinct data collector agents, creating new consolidated information.
* **Future value predictors;** which are implemented by using neural network structures designed for time series prediction. With this class of agents our intention is to implement entities that theoretically could predict future values in a time series. This feature is quite helpful in designing proactive management system.
* **Value classifiers**; which are implemented by using neural network as well, permit data classification in scales, like HIGH, NORMAL, or LOW instead of linear values. This classification is implemented by pattern machines and high adjustable being even possible to create neural network structures that adjust themselves based on past values of the time series.

So far the neural agents have been studied as an improvement for the system functionality. Neural agents work as servers for the neural network structures, where batches of data are submitted for processing and result returned. In the future we are idealizing to have hybrid agents, with both rules- based and neural structures, inside INTERFACE modules.

(c) Interface agents (AgI); are the bridges between the agent communities and the human managers. There are various possibilities for these interfaces, like e-mail interfaces, terminal like interfaces and web interfaces. Selecting which interface the human manager prefers the friendlier interface in some aspects are the terminal like interface, where humans can "chat" with agents using structured language similar to human's natural language.
Interface agents, like any other, have their functionality structured inside the community and also the agent world. When a new sentence is input, the local parser, which is implemented by means of rules inside the knowledge database, will try to translate and understand it. If this is not possible locally, usually due to lack of local knowledge, the task will be escalated a more skilled-inside the community or not- that will try to get translation for it. If in the end the agents can't still understand the human entered sentence it will reply-as of current implementation an "I can't understand you" sentence as ask to repeat.

(d) Facilitator agents (AgF); are communication management agents needed in the agent community to work as brokers for the information exchange process. They also implement some useful tasks related to data exchanging like list of available service names, routing, meditation and translation.

CONCLUSION

Managing the TOKENET local area network has been the aim of this study. The reason behind selecting this network type comes from the ability of this network to be used for a high speed local area computers network using bidirectional bus, all network separations which must be executed through the agents are based on the transmission and reception of control packets (token).

Tokens issued by the network and stations are used to schedule packets transmission in the from of a train on which a packet is admitted only after a reservation has been issued by the original station. The algorithm for the election of the end station, the recovery from failure, and the insertion of a new station are all based on special token transmission.The architecture of TOKENET is completely distributed: any station may be elected and station depending on which station is at some time connected to the network. When end station fail, replacements are elected among those stations that survived the failure.

The following will drive our future work:

To integrate even closer the promising field of neural networks inside current interface module, in a transparent and extensible way. So, furthermore, we will have truly hybrid agents that will be able to take advantage of the best of both words. For network management, we use a direct utilization by creating neural structures to predict temporal series dynamically. This would be used in proactive network management through future value prediction.

REFERENCES

1. CHikhouhou, M.M., P. Conti, K. Marcus, J. Labetoulle, 2000. A software agent architecture for network management: Case studied and experience gained. INSM Special Issue on Intelligent Agents for Telecommunications Management, 8: 3.
2. Ajmone Marson, M. and Gerla, 1983. TOKENET-A token based local area network. Proc. Mediterranean Electron Technical Conf., Athens,Greece.
3. Fratta, L., F. Borgonovo and F.A. Tobagi, 1981. The EXPRESS-NET: A local area communication network, integration voice and data. Proc. Intl. Conf. Performance of Data Communication Systems and their Applications, Paris, France.
4. Andrew S. Tanenbaum, 2003. Computer Networks. Fourth Edn., Pearson Education, Inc., Prentice Hall PTR, Upper Saddle River, New Jersey.
5. Maes, P., 1994. Agents that reduce work and information overload. Comm. ACM, 37: 7, U.S.A.
6. Stallings, W., 2000. Data and Computer Communications. 6th Edn. Upper Saddle River, N.J., Prentice Hall.