PARALLEL AND DISTRIBUTED DATA PROCESSING USING AUTONOMOUS SOFTWARE AGENTS

Ahmed Baita Garko
Department of Computer Science, Kano University of Science & Technology - Wudil
E-mail: abgarko@yahoo.com +234-806-4980-045
and
Gregory Maksha Wajiga and Mike O. Eegwurube
Department of Mathematics and Computer Science, Modibbo Adama University of Technology, Yola, Nigeria

Abstract
Some of the typical Communications Systems are the Global System for Mobile Communications (GSM), Code Division Multiple Access (CDMA) and Wireless Local and Wide Area Computer Networks that dominate the era of Information and Communications Technology (ICT). These systems (GSM, CDMA and ISPs) are both scalable in the sense that they are expandable. They are expandable by connecting more cities, towns and villages to their networks. Looking at the distributed nature of these networks, data is processed by remote login or Remote Procedure Calls (RPC), this causes congestion in the network bandwidth. This paper proposes a framework where software agents are assigned duties to be processing the distributed data concurrently and assembling the data for bills generation in a central server. The Agent-oriented software engineering was followed in coming up with the framework. The system efficiency and performance were improved as the data is processed concurrently and by software agents. In turn, a company maximizes its profit by saving a lot of money as software agents are not paid salary and allowances.

Keywords: Software agent, unified modeling language, agent-oriented software engineering

Introduction
Some of the typical Communications Systems are the Global System for Mobile Communications (GSM), Code Division Multiple Access (CDMA) and Wireless Local and Wide Area Computer Networks that dominate the era of Information and Communications Technology (ICT). These systems (GSM, CDMA and ISPs) are both scalable in the sense that they are expandable. They are expandable by connecting more cities, towns and villages to their networks. They are expandable by connecting more and more terminals (lines) or users thereby growing from hundreds to millions of subscribers.

Access / mobility of the billing data on the distributed systems
Looking at the complex and distributed nature of telecommunications equipment/devices for a particular nation or continent, and also the distributed nature of subscribers in various locations establishing connections and delivering messages from one location to another, it generates a lot of overhead on the side of the network usage. Consider the diagram for a typical data movement as it originates from the subscriber to the time it is processed in the Billing Department for invoicing.
Overview of software agent technology
According to Bradshaw (1996), since the beginning of recorded history, people have been fascinated with the idea of non-human agencies. Popular notions about androids, robots, cyborgs, and science fiction creatures permeate our culture, forming the unconscious backdrop against which software agents are perceived. Software agent technology has been a research area in Computer Science that deals with substituting human user by a group of computer programs that carry out the routine tasks autonomously with or without minimal human intervention.

What is an agent?
The Encarta Dictionary defines agent as a computer program that works automatically on routine tasks such as sorting Email or gathering information. A strategy white paper written for IBM offered the following definition of intelligent software agents:
Intelligent software agents are software entities that carry out some set of operations on behalf of a user or another program with some degree of independence or autonomy, and in so doing, employ some knowledge or representation of the user’s goal or desires.

Generally, agent is software that represents users in the same way the users would represent themselves. More definitions of agent can be found in (Bradshaw, 1996; Nwana, 1996; Griss, 2001; Dale, 1997; Rothermel and Schwehm, 1998; Samaras, 2005; Wooldridge and Jennings, 1995; Jennings, Sycara and Wooldridge, 1998; Lange, 1998; Jennings, 2000).
Why software agents?
There are a number of reasons why software agents are employed to carry out day-to-day (routine) activities that are centered on efficiency and reliability (Poy, 1996). Another important reason is the benefit to be derived by applying agent to carry out some complex routine operations, for the agents easier to maintain, platform independence and above all concurrent tasks delegation in a network environment (Dale, 1997; Maes, 1999; Jennings et al., 1998; Zambonelli et al., 2000).

Properties of an agent
Griss (2001), Nwana (1996), Wooldridge and Jennings (1995), and Poy (1996) suggested that not all pieces of software are regarded to be called agent; rather, for an agent to answer its name, it needs to have at least three of the following properties (Nwana, 1996). These are:

- **Autonomy**: The degree to which an agent is responsible for its own thread of control and can pursue its own goal largely independent of messages sent from other agents (Griss, 2001). According to Nwana (1996), Autonomy refers to the principle that agents can operate on their own without the need for human guidance (Huhns and Singh, 1998).

- **Cooperation**: For agents to cooperate they need to possess social ability, that is, capacity to interact with other agents or human users via a communication language (Wooldridge and Jennings, 1995; Nwana, 1996; Dickinson, 2006). That means it’s possible for agents to coordinate its actions without cooperation (Nwana et al., 1996).

- **Learning**: Agents usually learn to better assist its user in four ways (Maes, 1994)
  - By observing and imitating the user (i.e. learning from the user);
  - Through receiving positive and negative feedback from the user (learning from the user).

Typology of agents
Based on the listed properties in 1.2.3 above, Nwana (1996) suggested that software that has an intersection of at least two of the properties be regarded as an agent. The agent types are listed below as proposed by Nwana.

- **Collaborative agents**: These are agents that emphasize autonomy and cooperation with other agents in order to carry out any task (Nwana, 1996). These agents may learn but learning is not emphasized as a requirement for job execution.

- **Collaborative learning agents**: According to Nwana (1996), these are the same with collaborative agents, except that, learning is also emphasized apart from autonomy and cooperation in executing their tasks.

- **Interface agents**: This type of agents uses learning and autonomy in order to execute tasks for their owners. A good example of this kind of agent is just like a personal assistant who works with the user in the same work environment (Nwana, 1996; Maes, 1994; Dickinson, 2006)
Smart agents: these are agents that have all the characteristics of other agents; that is to say, they combine the behavior of interface agents, collaborative agents and collaborative learning agents (Nwana, 1996).

Statement of the problem
Most of the communications systems are distributed systems and the data that is generated on daily basis is originated from the network elements that are connected over a wide area network (WAN), usually there are number of switching systems that gather the data generated from the network elements for onward transmission to the billing system that used to be central for processing. The work of transporting the data generated from switches to the main billing center is done over the network using the remote logging facility of the distributed system by human users, this operation often used to have some problems like:

a. Breakage in the communication link between the client and the remote server
b. The remote server may be off without the knowledge of the human user
c. The data is subject to corruption as humans have direct access thereby creating the tendency of revenue leakages.

Aim and objectives:
The aim of the research is to design and develop a framework for software agents in order to carry out some of the routine tasks that are considered to be time, accuracy and mission critical in a distributed virtual environment for different locations on the network at the same time (in parallel).

The objectives are:

1. To explore the diverse operations carried out on routine basis in the Communications systems’ network and come up with the practical application of software agents executing these operations simultaneously.
2. To provide a framework that would be beneficial to both the customers and the Communications companies.

Related works
Some of the related works are:

a. Chira (2007) wrote similar thing on distributed computing but did not place emphasis on parallel processing of information and did not mention anything about heterogeneous nature of today’s distributed computing environment.

b. Dale (1997) also embarked on similar project but his emphasis was the internet, no special attention was paid in his work to the local area networks, also there was no emphasis on processing the information simultaneously.

c. Corchado, Tapia, and Bajo (2007) looks similar to my research but the architecture is designed for healthcare not communications systems and also no mention of parallel processing of information.

d. Manvi and Venkataram (2004) looks at the application of agent technology in communications from the telecommunication network management, but there was no mention of the kind of data generated and the manner the data is handled.

Methodology
Agent-oriented software engineering was employed in the design and development of the multi-agent architecture for distributed and parallel virtual environment. In designing the framework, Java Agent Development Environment (JADE) was used as it is within the Foundation of Intelligent Physical Agents (FIPA) standards. In some instances also, object-oriented technique was used to design and develop some components of the system especially at the remote location’s processes.

Parallel distributed processing using ANN
The entire telecommunications network together with its activities and architectures has so much resemblance with Artificial Neural Network (ANN). Considering the human neural network for example, neurons are located all over our body and they collect and send data from where they are located to the brain where the data is processed centrally, after which the output is generated and
communicated to the right actuator for the appropriate action to be taken.

Likewise for the telecommunications network and its billing system, the neurons are the various nodes (in this case mobile stations or handsets) originating billing data and eventually transmitting it to a central place called the billing system for processing in order to help the subscribers and telecommunications operators take appropriate action. A typical artificial neuron is represented as:

\[
\sum_{j=1}^{n} w_j x_j
\]

Where \( x_1, x_2, \ldots, x_n \) represent the input elements and \( w_1, w_2, \ldots, w_n \) the data being transmitted by the input elements.

The Output or activation function can be represented as:

\[
O = F (net) = F \left( \sum_{j=1}^{n} w_j x_j \right)
\]

1. Where \( w_j \) is the weight vector, in our case the data generated by the terminal \( x_j \). \( F (net) \) is referred to as the activation or transfer function. The variable \( net \) is defined as a scalar product of the weight and input vectors.

\[
net = w^T x = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n
\]

2. Where \( T \) is the transpose of the matrix, and, in the simplest case, the output value \( O \) is computed as:

\[
O = F (net) = \begin{cases} 1 & \text{if } w^T x \geq \theta \\ 0 & \text{otherwise} \end{cases}
\]

3. Where \( \theta \) is called the threshold level.

From the equations 1 to 3 above, we used the ANN as a basis for solving our problem in the pre-billing activities. Another important reason is that, ANN has the learning ability. The software agents need to learn in order to identify the right data for the postpaid billing, and the agents should also be able to differentiate this data from the Interconnect billing data.

**Feed-forward network**

According to Rojas (1996), ANN is used in many cases as a black box: a certain input should produce a desired output, but how the network achieves this result is left to a self-organizing process.

As the software agents are carrying out the pre-billing activity of parallel distributed processing and sending the processed data to the central billing system for final processing over the network, then a particular shortest path algorithm that is more efficient is selected and used by the software agents for the data transmission from a particular location to the billing system.
Learning behavior of the software agents
Knowing that for any software to be regarded as agent it has to have the following properties:
  ✓ Autonomy
  ✓ Cooperation
  ✓ Learning
The learning type we proposed for this type of application is unsupervised and reinforcement learning. This is to ensure more autonomy as the software agents should learn on their own, without any human interventions.

Agent-based postpaid billing system

Agent system design
After coming up with the required number of agents to be used in our Agent-based Postpaid Billing System, and after identifying the different activities to be executed before a bill is generated and sent to the right customer, it is equally important to depict the relationships between the agents (as actors) and the processes as use-cases on the other hand. The UML use-case diagram for our Agent-based system is shown in figure 5.

Agent-based postpaid billing system

![UML Use-Case Diagram for the Agent-based Postpaid Billing System](image)

Figure 5 UML Use-Case Diagram for the Agent-based Postpaid Billing System.

Activity diagram for the postpaid billing system
Another diagram of interest to this research is the activity diagram. This diagram shows the procedural flow of control between two or more agents while executing an activity. The UML activity diagram for our proposed Billing system is fully described as shown in figure 6.
Organizational structure of agents in our proposed system
In this research the agent’s class structure presents two main types of agents; agents that transport data (file) from one location to the other over the distributed system after carrying out one job or another, and other group of agents that execute a particular function without having to transport any data. This is because in the Pre-billing sub-system, after the payments and CDR processing, there is need for the two agents to transmit their processed file(s) to the main billing system. While the BillGenerator and Reportgenerator just execute their functions with no transfer of any data. The BillDispatcher basically performs data or information transfer as its function. But all the agents are static in nature. We described the agent’s structure by starting from the top of the hierarchy, where all the types of agents are rooted. The hierarchy for the agent organizational structure is shown in figure 7. All the classes are situated at the agent package.
Every agent on the structure in figure 7 is designed to be running as a thread software or program just to ensure parallel or concurrent processing on our proposed Billing System. To achieve this, an interface called Agent was declared first to extend the `java.lang.Runnable` interface. By this extension, all the subclasses of the Agent interface are automatically threads. Since the Agent interface contains all the necessary definitions for the methods that qualify a class to be an agent, then it was at the top of the hierarchy with the subclasses inheriting from it.

Looking at the structure in figure 7, we have two types of agents, the transport and non transport agents. That is, agents that transmit data/information to another location on the proposed system, and those agents that do not transmit anything respectively. Both types of agents were declared to be abstract classes that implemented the Agent interface. In our structure, these types of agents are related to communicators that enable them to send message(s) to each other. Communication is one vital role exhibited by our software agents to collaborate in order to achieve their various goals.

The transport and non transport agents are both static in nature, meaning that they don’t move around, they are permanently on fixed computer system. Each individual agent resides on one place, interacts with its environment (server and other agents) to perform its duties. Mobility is just one
characteristic out of many that qualify software to be called an Agent.

The transport agents are the types that execute their functions and in addition transport one or more files to another location (machine) on the distributed system. There are three (3) agents namely: *CDRProcessor*, *PayProcessor*, and *BillDispatcher* that belong to this group (class). This class was declared to extend the Agent class. The individual agents in this class are each declared to extend the transport agent. All the agents in our structure are reactive and autonomous in nature.

The non transport agent on the other hand don’t transmit any data, rather they are stationed in one place and they execute their functions autonomously and reporting or communicating the outcomes of their execution to either human users (Billing Administrators) or other software agents. There are two agents namely: *BillGenerator* and *ReportGenerator*. These agents were both declared to extend the non transport agent abstract class. These agents are always reactive in nature.

In addition, the transport agents as they send data from one machine to the other autonomously implemented the *java.io.Serializable* interface in order to enable them to be serialized. Before sending data to any machine, the agent under this group requests the server where it is currently executing to transport the data to the required machine. In turn the server makes the request on behalf of the agent, and then performs the transport to the required machine provided the request is granted. The transport agent class is related to the data class, which keeps the data the agent wants to transmit, and also the authenticate class, which is used for the purpose of authentication by either the agents or the requesting server. All the agents in this group were declared as final to prohibit extension, as fraud is one of the areas we are addressing in this research.

**Design of the agents communication system**

In our design, the agents system’s communication facilities are made inbuilt with the agents and the servers these agents are residing. Looking at the distributed nature of our Billing System, the communication can either be a local communication within a particular system (server) or remotely over the network, depending on the object needing to communicate. In our Agent-based system, the environment for the agents’ interactions presented three types of objects among which are: the agents themselves, the servers where the agents reside and finally the users that carry out some other operations in the Billing System.

It should be categorically stated that, the Billing Administrators get access to the Billing system through the local graphical user interface provided for human users. This is to ensure security of both the software agents and their execution environments.

Since we have identified three (3) communication objects and the communication is always one-to-one, we have five (5) possible communication patterns that can arise from these objects in our proposed system. These communication patterns are:

- **Agent/server communication**: This type of communication pattern must be done locally, that is, the agent wishing to embark on this type of communication must be residing on that particular server. A typical communication of this nature is a local procedure or method invocation where there procedure or method and agent live on the same machine.

- **Agent/agent communication**: The communication here is either local or remote. If the agents are residing on the same machine as in the case of *CDRProcessor* and *PayProcessor* agents, the communication is local. A particular agent can send a message to another agent on another server using message passing interface; this type is remote communication.

- **User/agent communication**: This type of communication pattern is possible, it is achieved via server. A human user wishing to communicate with an agent does so by sending the message to the
server which in turn sends it to the agent’s CA. For this communication to occur, the user and agent must be on the same server, meaning that the communication is purely localized, as remote communication cannot be achieved according to our design.

☑ Server/server communication: This type of communication is carried out remotely, where a particular message is sent from one server to another over the network. The main protocol that is used in this type of communication is the remote method invocation (RMI).

☑ User/server communication: As earlier pointed out that apart from the software agents, there are some human users (Billing Administrators) that perform some functions in our proposed system. The communication between a human user and a server is achieved by using local interfaces either provided by the operating system of that server or by the use of local graphical user interfaces on the Billing System.

Having identified the two (2) main types of communication as remote and local communications that exist among the objects (agents inclusive) that interact in our proposed system, therefore, we defined two interfaces that made our derived classes to be able to communicate either remotely or locally. For the purpose of understanding we call them RemoteCommunicator and LocalCommunicator for the remote and local communications respectively. The LocalCommunicator defines all the local communications between the objects in our system, therefore, agent-server, agent-agent and user-server interfaces were derived from it. The communicator’s class structure is shown in figure 8 and also the classes are contained in the communicator package.

Basically, in our system we have four communication patterns that are agent-agent in nature. These are:

- **CDRProcessor** – **BillGenerator**
  - Remote
- **PayProcessor** – **BillGenerator**
- **BillGenerator** – **ReportGenerator**
  - Local
- **BillGenerator** – **BillDispatcher**

The design for the communicators as they affect our system based on the agent-agent communication pattern is shown in figure 8.
As can be seen in figure 8, four derived classes, two for remote communications (CDRProcessor – BillGenerator and PayProcessor – BillGenerator) were defined to implement the AgentServerCommunicator, while the other two classes (BillGenerator – BillDispatcher and BillGenerator – ReportGenerator) were defined to implement the AgentAgentCommunicator. The AgentServerCommunicator and the AgentAgentCommunicator abstract classes are related to the Authenticate class, this is to make sure that whenever communication is to occur or take place, there must be a check point to have permission. This is just to minimize fraud and ensure system security.

System implementation
Having come up with the various designs of the components of our proposed system, for the results to be obtained and evaluated, the designs were implemented. In the implementation stage, client-server network architecture was used with four computer systems, one system as the server, while the remaining three as the clients or workstations.
Software agents reside on both clients and the server depending on the functionality of the agent. *Payprocessor* and *CDRprocessor* reside on the clients, while *BillGenerator*, *ReportGenerator*, and *BillDispatcher* reside on the server. Java was used to implement the agent system designs, while Visual Basic 6.0 was used to implement all the processes.

**System requirements**
In carrying out the system implementation, some components in form of hardware and software were required. The experimental environment used was comprised of a local area network (LAN) with four computer systems. The following are what we considered to be the minimum requirements as far as the billing system is concerned, considering the volume of data that is involved in billing activities.

**Hardware requirements**
As mentioned earlier, four computer systems were used to implement our proposed system. The hardware specifications are as stated below:

<table>
<thead>
<tr>
<th>Computer</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer1</td>
<td>Hp tablet pc</td>
</tr>
<tr>
<td>HDD</td>
<td>250GB</td>
</tr>
<tr>
<td>Processor</td>
<td>AMD Turion 2.00GHz</td>
</tr>
<tr>
<td>RAM</td>
<td>2.00GB</td>
</tr>
<tr>
<td>Role</td>
<td>Operational system server, localhost server</td>
</tr>
</tbody>
</table>

| Computer2  | Omatek note book                   |
| HDD        | 250GB                              |
| Processor  | Intel Atom 1.66GHz                 |
| RAM        | 2.00GB                             |
| Role       | client1                            |

| Computer3  | Speed star                         |
| HDD        | 250GB                              |
| Processor  | Pentium Dual-core 2.10GHz          |
| RAM        | 2.00GB                             |
| Role       | client2                            |

In addition to these computer systems highlighted above, in conducting the experiment, LAN was set up using a 16-port D-Link switch and UTP cat 5e cable in a star network topology.

**Software requirements**
Having highlighted on the hardware requirements of our new system the remaining component that manipulates the hardware is the software. The software requirements for our new system are as follows:

- Visual Basic 6.0 was used to implement almost all the processes from CDR and payment processing to bills generation and distribution.
- Mysql RDBMS was used as our back-end to develop and manage all our database tables

**Results and discussion**
Reports are results of any data processing system that involved collection of data and processing it by applying some techniques. The result of data processing is information that is used by managers for decision making in any organization. Below are some sample reports that are generated at the end of a billing cycle. Contrary to what is obtained currently, the reports are in softcopy form since everything is paperless.
Discussion: The results obtained from implementing this framework have shown
improvement in both the performance and efficiency of the data processing system. The time taken to be processing data remotely with some level of risk of network failure and bandwidth congestion were eliminated, as the software agents are residing on the clients executing their tasks with or without network connection.

Future work

Application of mobility to the software agents: In future, the framework can be modified to allow the software agents to move around. Currently, in this framework, the software agents are static in nature; the security threat is minimized by making them static and only run able in their own various allocated places.

Security of the software agents: By making the software agents to be mobile, then makes them vulnerable for attack, this opens another area of future work; thus, the security of the agents as they move around on the network.

Reference


Dale J. (1997), A mobile agent architecture for distributed information management, a PhD thesis, University of Southampton, United Kingdom pp 38-98


