A Multi–agent System for Dynamic Network Reconfiguration

Artur Maj, Jarosław Jurowicz, Jarosław Koźlak, Krzysztof Cetnarowicz

Department of Computer Science, University of Mining and Metallurgy
Cracow, Poland
arturamaj@poczta.onet.pl, jurowicz@poczta.fm,
kozlak@agh.edu.pl, cetnar@agh.edu.pl

Abstract. This paper presents a system for dynamic network reconfiguration based on intelligent agents and market–oriented methods. Reconfiguration encompasses routing changes as well as building new and removing unused connections between network nodes. The paper presents a short review of research background in network management methodologies, a description of the system's model and an overview of experimental results

1 Introduction

The number of Internet users and computers doubles every year. Recently, popular multimedia applications have stimulated the growth of the amount of data sent between the machines. Internet providers and e-commerce organizations are interested in the network quality improvement. Buying more bandwidth has usually solved this problem up to now. An alternative solution is to manage effectively the resources already possessed. The basic feature required from new generation networks is a *dynamic adaptation* to changing conditions, and it means:

- load optimization for making use of full potential;
- automation of configuration and administration processes;
- greater scalability;
- autonomous reaction in case of a failure or congestion.

Currently, most network management decisions needs participation of the man. This situation has the following disadvantageous consequences:

- data flow paths do not have to be optimal;
- centralization makes the „bottleneck" effect more likely;
- fixed routing is not applicable in case of breakdown;
- problems with management of large–scale networks;
- human factor is unreliable.

To some extent, centralized management of networks is in defiance of their nature – packet networks are decentralized. Regretfully, there are few methodologies taking this into account.

Multi-agent systems researches bring a really new quality to telecommunication networks management. A mobile agent can move thorough the network, gather the information on its state, exchange it with other agents met on the way and make changes to the network configuration. During the decision-making process, intelligent agents are able to analyze amounts of data much bigger than the man could take into account. Finally, one can obtain a system automatically self–adjusting to the load distribution and reacting instantly to faults and disturbances. A well–organized multi–agent system uses only local data; there is no need for global synchronization of decisions.

An effective method of intelligent agents coordination is market–oriented approach. Its usage brings three basic advantages: information propagation, optimization and opportunity to build a billing system for network services, costs and return of investment estimation and other market activities.

2 Research Overview

2.1 Network Management Methodologies

Network management systems (NMS) can be divided into a few general classes (after [3]):

- **centralized NMS** – there is one management interface to a whole network. Administration commands are mapped directly on protocol operations. This class includes commonly used SNMP.
- **weakly distributed hierarchical NMS** – agents installed on managed network devices use a predefined command set. This class consists of CMIP, RMON, SNMPv2.
- **strongly distributed hierarchical NMS** – decisions are made by an active and partly autonomous agent residing on a network device – and this reduces the load on the management station. This class includes active networks, mobile code as well as distributed objects systems.

  The basic idea of mobile code systems is execution of some computer program to the network agent set on a network device. *„Active networks enable their users to put custom programs into the network nodes."* [5]. These programs can process a data stream and alter it in some way (e.g. encrypt or compress). This idea breaks the sanctity of the higher-level data stream. Routing is not static, but it depends on the code execution result.

- **cooperative NMS** – based on using intelligent agents. In the cooperative approach, a manager gives an agent the problem to solve. The agent's task is to choose and execute proper actions which will lead to the solution. Agents move freely in the network, and thus the infrastructure has to be equipped with suitable mechanisms.

Market–based methods are successfully used in the network optimization. It is based on the assumption that microeconomic rules will lead the system to the equilibrium, which will be globally optimal, i.e.:

- resources are distributed accordingly to the requirements in different system areas;
- the most profitable configuration is chosen.

Market – based methods do not need a centralized management, all decisions are made on a basis of local knowledge. An agent makes decisions which maximize its profit. The competition between agents leads to self-regulation of the whole system [6].

2.2 Dynamic Network Management Systems

In [4], Minar, Kramer and Maes presented a dynamic routing management model using mobile intelligent agents.

The goal of agents, circulating in the network, is gathering information about existing connections and entering this information in routing tables on the network nodes. The measure of algorithm efficiency was the percentage of nodes having valid route entries vs. the one of the hosts designed.

The average efficiency grew rapidly to 0.8 at the 40th iteration, and then was oscillating between 0.73 and 0.84. The agents population can ensure a relatively high connection level in a dynamic changing network, and performs it in a very stable manner.

The most influencing parameter is the next node choice algorithm: when agents were choosing the most recently visited one, the average efficiency was 0.792; the system with agents traveling randomly has an efficiency value of 0.646. It is not any disqualifying difference – random agents can be useful in rapidly changing systems.

The system presented in [2] by Jennings, Gibney, Vriend and Griffiths manages the telecommunication network using intelligent agents and market–oriented methods. It consists of three layers: network infrastructure, multi–agent system responsible for routing and end–user interface. In the system there are three agent classes: link agents, path agents and call agents. Agents work on two markets: link market and path market. Offers and bids are matched by using the sealed bid auction mechanism.

Results of using a static routing are almost the same as those when using the market–based method (94,8 % and 95,4 % connections set together, respectively). More than a half of all calls were set by using the shortest path, and only 14 % by using the least effective way.

The authors created a routing mechanism, which can successfully compete with traditional static systems without centralized management. Thanks to load prediction the service is fast. Routing and load balancing algorithms are easy to implement. The system allows an instant calculation of the bill for a network service.

3 System Conception

Our goal was to create a computer-aided model of a telecommunication network which could get adapted without any external intervention to current traffic conditions through automatic routing reconfiguration and by adding and removing connections between the nodes. The intelligent agents have to analyze the network state and make the configuration changes. The system self-regulation is based on the microeconomic mechanisms.

To make our model more efficient and flexible, we gave up pricing at the most elementary level (as it is in [2]). Price changes on the basis of network traffic statistics are made by agents.

We also abandoned negotiations which would complicate and lengthen the routing process. We did not use auctions, because they usually give some number of “losers”. In our model there must be no packets which cannot reach the target because of lack of money or get there with a significant delay.

The starting point in our work was the system described in [1]. From it we took the network architecture and packet movement basics. The major changes were made in the traffic organization and market rules. In [1], packets are directed via the shortest path – in our system they go by using the shortest path they can afford – when they lose all the money they go via the cheapest route. After reaching the target, the packet pays for the route beginning from the cheapest connection to the most expensive one. Each port has its own “accounting” for profit estimation. Incomes much lower than expected signal that the connection is too expensive. This solution is good for a simulation, but it does not allow to create a complete macroeconomic environment.

Agents work on behalf of the mother nodes and report only to them the information gathered.

The only departure from the distributed paradigm was in the creation of the centralized bank. In the real network, there should be at least one bank network, if not more.

The system was realized as a simulator. At the input, simulation regards the entry network and data streams definitions. The program imitates operations in the real network: packets and agents move, the network gets reconfigured. As an output, the simulator produces a new network structure, optimized to the given load.

4 The Model

4.1 Network Architecture

Network hierarchy. The network consist of two basic elements: host (alternatively called node) and line (connection between two adjacent nodes). Hosts are grouped in subnets. Each subnet has one host appointed as a gateway to the other subnets.

For the sake of standardization we troduced the higher-level subnet consisting of hosts representing subnets. E.g. host “BB” is directly connected with host B1, which is the gateway of subnet B and both hosts function as the two–interface router.

Hosts. In our model the term „host" means miscellaneous network entities: network server – data source; client computer; router.

Basic events and operations effected by host are: generation and routing of packets, agent emission, own configuration changes, building new and destroying unused connections.

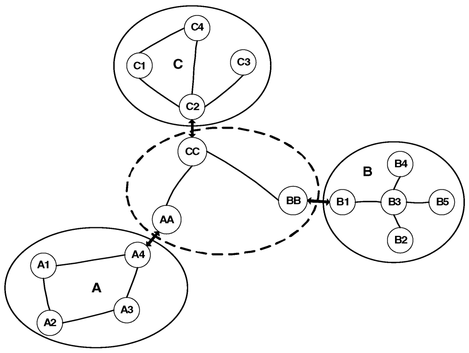

Fig. 1. Network architecture

Ports and lines. Hosts communicate by using lines, anchored in ports. Each port has two buffers: IN and OUT. The output buffer size is a few times greater than the line capacity and it has to prevent against short–time congestions. All traffic goes through the hosts interior, where the routing mechanism is. In a port there are also two separate buffers for agents. Each port charges packets coming out to the line. A port has two statistical agents collecting the data about packets traffic in and out.

The outgoing line has the following parameters: capacity (packets per simulation step), price for single packet transfer; length; maintenance cost and guaranteed capacity (it is the protection against network fragmentation). The decision on changing the line capacity or building a new line is made by a host.

Routing table. In each host there is a routing table. It includes entries about all adjacent hosts and some other hosts in the network.

The single record in the routing table looks as follows:

Destination address	*Port*	*Time*	*Cost*	*Distance*	*Frequency*

For the one host there can exist a few entries in the routing table differing from each other in time or cost. Beside specific records, there are default entries, consisting only of three fields: *source port, destination port and frequency*. All modifications of the routing table are made by agents specialized in network search and data stream analysis.

Packet traffic. Packets are generated in the hosts according to the specification given as a simulator parameter. Each host produces some amount of packets to some other host. The packet has the following attributes: a destination host identifier, a money amount, the history of a covered way and TTL – decremented with each line passed.

Passing the line, a packet writes in the way history - a record - which consists of the following fields: price, line length (geometric), bank account number to which it has to pay for a line passing. As the packet has reached the destination, it pays for the travel in the order from the cheapest to the most expensive one. It means that sections relatively more expensive are more likely to be unpaid.

While moving from one subnet to another, the packet does not enter any subnets except the destination subnet. When the destination and source hosts are both in the same subnet, the packet does not leave this one subnet.

In case the packet's TTL reaches 0, the packet is deleted. It is possible when there is a full input buffer on the port or when the routing is not configured correctly.

4.2 Static Agents

Main agent. Each host owns its main agent responsible for creating and updating the routing table, financial control, building new and deleting unprofitable lines. The main agent runs once per simulation tour. Once per 50 simulation tours main agent checks whether outgoing links are profitable. If a link is not profitable, its capacity is changed to [average traffic] * 1.3 or to guaranteed capacity, if it is bigger than proposed capacity. If proposed capacity equals 0, the connection is removed.

A statistical agent leads short–and long–term statistics of the packet's traffic. Agents of this type reside in each port and each host. A statistical agent takes information about the source and destination from every packet running in the network.

Pricing agent. Each port has its own resident pricing agent, which manipulates the price of the outgoing line. The agent makes adjustments to the price and looks for the most profitable value. The algorithm is written in such a way, that the price approaches the optimum logarithmically: changes are initially big, and then lower to the optimal price.

4.3 Mobile Agents

The task of **the seeking agent** is to find a route to the specific host in the network. During its way, the agent keeps choosing the successive port randomly until it reaches the destination or finds in the routing table of other host an entry about the host searched. The report for the mother host includes: destination host address, travel time (number of hops); route length; route cost; identifier of the port through which the seeking agent left the current host.

Checking agent. The goal of the checking agent is verification of routing table entries. It moves through the network according to the routing table and reports any changes.

The tracing agent observes routes on which packets move. This agent is sent to the specific host, like the seeking agent. The only difference is that it does not draw the next port, but looks into the routing table and goes the same way as the packets stream. Decision to which hosts tracing agents have to be sent is made on the basis of information collected by the statistical agent.

The marketing agent seeks for the nodes to which a new connection could be worth building. It follows the biggest stream of packets outgoing from his mother host and gathers statistics on all hosts visited. When agent reaches the end of the packet stream it begins to analyze the data. Agent estimates the best capacity, profits expected and cost of building new links from the mother host to all hosts visited. Additionally expected loss of profit on other connections is calculated and taken into account. Then the most profitable proposition is chosen from among affordable ones. Parameters of the connection to be built are sent to the main agent, which builds a new link and modifies the routing table.

Mobile agents work on behalf of their mother hosts – it comes from the basic idea of competition between hosts and agents. Mobile agents do not exchange gathered data between each other directly and do not cooperate in any other manner. They only modify routing and statistics tables of their mother hosts. Other agents can read these tables freely. It is the way the information propagates through the system.

The bank has two roles in our system. The first one is a common bank: storing information about accounts of all nodes and making all essential accounting. The other one is charging taxes for building and operation connections. It introduces the cost element in the system.

4.4 Network Simulator algorithm

The model analyses the network status on the ends of discrete short time brackets. During a round we can distinguish three phases characterized by the types of operations made.

Phase 1 – packet processing

1. New packets are generated in the host.
2. Host takes incoming packets from all its ports and sends them further accordingly to the routing rules.
3. Main agent in each host is running its code. Beside other activities, it generates new mobile agents.

Phase 2 – mobile agents processing

1. In each host, mobile agents are moved from all ports to the host’s internal buffer.
2. Each agent is activated. It takes from host the information needed and processes it (or can send it to its mother node).
3. An agent is sent to the chosen port or stays in the current host.
4. Removal of the agents, which have finished their mission, from the system.

Phase 3 – packets and agents moves

1. Each port sends to the line packets from its output buffer – but no more than the line capacity allows. If there are more packets, the buffer works as the FIFO queue. Packets come up to the input buffer in the port at the other end of the line.
2. Then, all mobile agents are sent to the desired port – there are no capacity limitations.

5 Implementation

The functional project of the system was made by using UML. Then, there were created object classes corresponding to network architecture elements (host, port, line, packet, bank), mobile and static agents. Interactions between different classes were defined, too.

On the basis of this project, the program was realized in C++. Microsoft Visual C++ 6.0 was used as a compiler and developer environment. All the code has been written according to the ANSI C++ standard, which allows to adapt simply the work under different operating systems.

6 Results of Research

Our research on the system was made in two general areas: network traffic management and rebuilding network connections. Experiments were made by using two networks: network A was a one–piece local subnet consisting of 32 hosts. The other one (network B) was built from three local networks.

6.1 Network Traffic Management

We were interested in the analysis of time needed to reach the stable state by routing table and in the level accuracy of tables. The experiment was led with an option of building new and removing unused connections turned off. The network links bandwidth was set to the value so high that congestions should not occur. We carried out the experiment by using different routing table lengths. The percentage of packets reaching the destination was taken as an efficiency factor.

For the network A, the required routing table size was at least 8. At this value, the percentage of missed packets was about 1%. This value settles after 430 rounds. With the routing table size of 14, the efficiency grows to 99% almost instantly. That big routing table is required only for few key hosts, which transfer majority of traffic. We observed that the efficiency fell to 90 % after 840 rounds – it can be caused by periodic default gateway value changes.

Comparing results from networks A and B, we can remark how dividing the network into subnets improves the efficiency. It simplifies the routing and allows to reduce significantly. the routing table size. In the network B, when the routing table size is 0, the efficiency instantly reaches 90%; with size 3 it is 98%, and from 8 onwards it is 100%. This is caused by the fact, that when the routing table size is big, almost all hosts are in it, and packets do not use default gateway entries.

6.2 Network Structure Rebuilding

Network structure rebuilding experiments were made only on network A. At the beginning, all links have the bandwidth set to 300, and a guaranteed bandwidth set to 10, routing table size is 20, amount of money per packet is 5000. After 500 rounds the routing tables are settled, the traffic is often much lower than a bandwidth and profits do not surpass the costs. Of course, it is not optimal. We set the amount of money per packet to 1000. After 2500 rounds, the bandwidth is reduced almost on all connections.

After a fivefold increase of money amount per packet, 4 new links appear on relatively short and loaded routes. The fee for the link maintenance is in proportion to the square of the link's length, so the link cannot be too long. On the other hand, the static base fee prevents from creating too short links.

After increasing the amount of money per packet to 6000, and after 4500 simulator rounds, the topology of network graph was almost identical. It is by all means correct – it proves that the system runs stably. We expected some changes to be visible after a significant increase of money in the system. To check what the character of these changes would be like, we set the packet money amount to 10000. After 4500 rounds, 9 new links were built, even on long distances (they are marked with black solid arrows on figure 2.). The capacity of almost all other connections was reduced. Only few links remained intact.

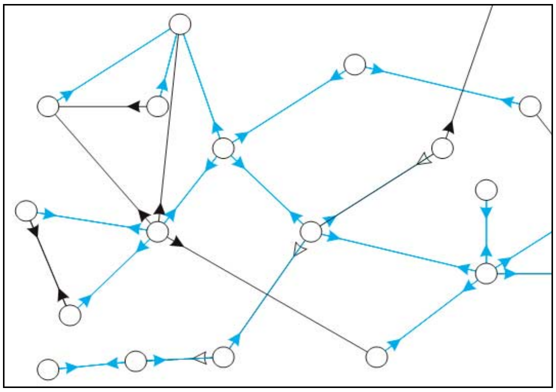

Fig. 2. The network fragment – packets traffic

All new links are profitable; in a few cases profits strongly exceed the costs, and this may makes one worry if the system works properly. On the other hand, new links are not built between all hosts. An other advantage, which cannot be read from figure 2, is the fact that the network shape settled between the 2000th and 2500th round and during last 2000 rounds; there were not any changes in it. It can attest to the system stability.

In the last experiment, the bandwidth on all links is set to 10, and this causes that the network is completely congested. The expected behavior should be similar to building the network from the basis. The money per packet amount is set to 10000.

In spite of similar initial values (except the connections bandwidth) like in the previous experiment, the number of new links is much greater than in the previous simulation. Marketing agents prefer to build new connections than to increase the bandwidth of the existing ones – in this case an agent does not take into account the loss of profits on the old connection.

6.3 Critical Remarks

Experimental results prove that system works according to assumptions in the area of routing management and network structure rebuilding. We should reconsider the way the new connections are built for making this process more flexible.

We found empirically that the routing systems cause chaotic creation and deletion of links in certain specific conditions.

The system should be optimized for computational efficiency. The standard simulation (5000 rounds) takes about 10 minutes on the PC computer with 600 MHz processor. It makes an effective testing hard to be carried out.

7 Conclusions

While designing our system, we referred to similar existing systems. The network architecture model is based on the one described in [1], but we have modified deeply the market elements of the system. The economical background was to achieve an optimal balance between computational efficiency and flexibility.

The simulator can be used to test various agent decision algorithms and marketing mechanisms in the research for the optimal self–configuring network. The side effects of the simulator work are optimally configured networks, and thus the system can be used in conventional network design.

The main values of our system are: proper modeling of the real telecommunications networks, possibility to change many parameters of the system configuration. Our system is of an innovative character.

Undeniable disadvantages are: huge computational complexity of the simulator, lack of the graphical interface, necessity of parameters adjusting by hand depending on the type of the network.

Our system in present form is complete, but it does not mean it cannot be expanded. These are a few most important changes possible to introduce:

- building links between any hosts from different subnets;
- coalitions of agents sharing a budget and goal;
- data trading, data mirroring;
- rebuilding charge system (for billing purposes).

References

[1] Krzysztof Cetnarowicz, Małgorzata Żabińska, Ewa Cetnarowicz. A Decentralized Multi-Agent System for Computer Network Reconfiguration.

[2] M.A. Gibney, N.R. Jennings, N.J. Vriend, J.M. Griffiths. *Market-Based Call Routing in Telecommunications Networks using Adpative Pricing and Real Bidding*. In proceedings of the 1999 Conference on Intelligent Agents for Telecommunications Applications, Stockholm, 1999.

[3] Jean-Philippe Martin-Flatin, Simon Znaty. *A Survey of Distributed Network and Systems Management Paradigms*. EPFL, SSC, Lausanne, Switzerland http://sscwww.epfl.ch, 1998.

[4] Nelson Minar, Kwindla Hultman Kramer, Pattie Maes. *Cooperating Mobile Agents for Dynamic Network Routing*. Mobile Computing and Communications Review, tom 3(2), 1999.

[5] D.L. Tennenhouse, D. Wetherall. *Towards an Active Network Architecture*. ACM Computer Communication Review, 26(2):5-18, 1996.

[6] Nir Vulkan, Chris Preist. *Automated Trading in Agents-based Markets for Communication Bandwidth*. http://www.hpl.hp.com/techreports/2000/HPL-2000-24.html, 2000.