# Automated Allocation of Multi-provider Service Demands

Monique Calisti[1] and Boi Faltings[1]

Laboratoire d'Intelligence Artificielle
Swiss Federal Institute of Technology (EPFL)
CH-1015 Lausanne, Switzerland.
{calisti,faltings}@lia.di.epfl.ch

**Abstract.** The increasing number of competitors and the growing traffic demand are the main factors pushing for a more dynamic and flexible service demand allocation mechanism. Software tools are becoming fundamental for supporting human decisions and/or for reducing the need of human intervention. Agent technology promises support for *pro-active* and *autonomous* network control that would enable the automation of many network provider tasks. In order to prove the feasibility of such automation, a multi-agent simulator for the allocation of service demands has been developed. This paper describes the simulator and aims to give a useful feedback for agent developers.

## 1 Introduction

For many distributed applications requiring intelligent, autonomous and communicating software entities the multi-agent technology seems to be on of the most promising answers. Among others, the networking community is also investigating the deployment of autonomous agents for different purposes: network control, network management, resource allocation, etc. (see [8] for a good collection of papers). In particular, in a multi-provider environment the distribution and the heterogeneity of actors, resources and technologies suggests a management solution based on static and/or mobile distributed software entities. Such autonomous entities would have the ability to directly invoke effective changes to switch and router controllers, without the intervention of human operators [12]. For evolving the interaction between distinct network providers the *Network Provider Interoperability-multi agent system* (NPI-mas) has been developed [2]. The basic idea is to provide an efficient and flexible mechanism for self-interested agents representing different network operators for the allocation of service demands spanning distinct networks. The overall service provisioning process has been modelled has a *Distributed Constrained Satisfaction problem* (DCSP). Constraint satisfaction is a powerful and extensively used Artificial Intelligence paradigm [11] that involves finding values for problem variables subject to restrictions (constraints) on which combinations of values are acceptable. The multi-provider service demand allocation can be considered as a DCSP [14], since the variables

C. Linnhoff-Popien and H.-G. Hegering (Eds.): USM 2000, LNCS 1890, pp. 138–149, 2000.

are distributed among agents and since constraints exist among them. This paper focuses on the development of the NPI-mas simulator, on the description of its software components and on first simulation results with the purpose of:

- Evaluating the feasibility of the DCSP algorithms developed for supporting the automatic allocation of multi-domain service demands. The simulations aim not only to validate the algorithms, but also to quantitatively estimate the performance that could be obtained in specific network scenarios.
- Sharing with the readers the experience gained by programming distributed, autonomous and communicating software entities.
- Discussing the potential of this multi-agent paradigm and the major limits for its integration in a real network.

Section 2 describes the architecture of the NPI-mas simulator its main structural components. Quantitative results about the NPI-mas algorithms and an evaluation of the simulator are presented in Section 3. Section 4 comments on future work and concludes the paper.

## 2 An Agent-Based Simulator for Service Provisioning

NPI-mas has been conceived and developed as a virtual place that allows the simulation of multi-domain service demands allocations. A service demand is defined as $d_k ::= (x_k, y_k, qos_{req,k})$, where $x_k$ is the source node, $y_k$ the destination node, and $qos_{req,k}$ the required Quality of Service (QoS). A demand may be anything from a video-conference to a virtual link in a Virtual Private Network. In our framework, the QoS requirements correspond to *bandwidth* and *end-to-end delay*. The 3 types of agents populating NPI-mas are End User Agents

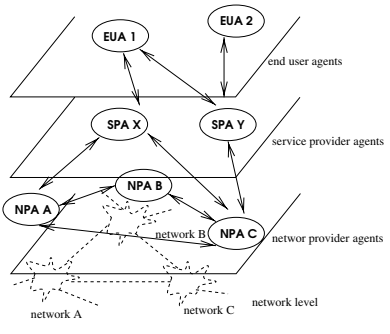


**Fig. 1.** *The agents' tribe populating the NPI-mas simulator.*

(EUAs), Service Provider Agents (SPAs) and Network Provider Agents (NPAs) (Figure 1). An EUA acts on behalf of a final end user, expressing his needs and formulating service demands. An SPA processes service demands and contacts

NPAs that own or directly control specific network resources. An NPA contacts peer-operators, whenever interactions are needed, e.g., when a service demand spans several networks.

### 2.1 The NPI-Mas Architecture

The NPI-mas simulator is entirely implemented in Java. JATLite [1] has been adopted as the main framework for the agent development. The main components of the simulator are:

- The *router* that receives messages from all registered agents and routes them to the correct recipient.
- The *Agent Management Interface* (AMI) that is a graphical interface which allows the selection of a multi-domain scenario, the creation of agents, and the visualisation of simulations' outputs.
- The *agents' community*. Several EUAs and SPAs can be created. A distinct NPA is associated with every network provider in the simulated scenario.

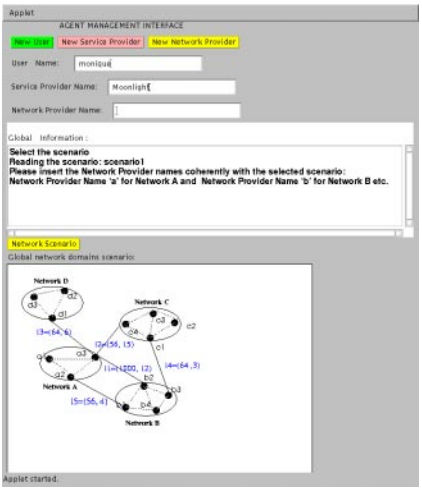


**Fig. 2.** *The Agent Management Interface.*

First of all, the AMI (Figure 2) enables the selection of a randomly generated network environment and the creation of all the agents populating it. Each agent automatically registers its identity with the *router*. In addition, every NPA recovers the data describing the network topology from dedicated *management information databases*. From the AMI, a text window displays the simulation

[1] JATLite is a set of packages that facilitate the agent framework development using Java. On-line documentation can be found at: http://piano.stanford.edu/

outputs such as the computed end-to-end paths from the source to the destination network, the results of intermediate steps of the demand allocation process and final negotiation outcomes.

Every NPI-mas agent is associated with a graphical interface that displays all messages that are sent and received by the agent. Furthermore, from the interface associated with an EUA, it is possible to enter a specific service demand in terms of source and destination nodes, required amount of bandwidth, available budget, required end-to-end delay and temporal constraints. Next, the EUA sends a *call-for-proposal* message to a selected SPA. An SPA represents both the *client* of the network services offered by the NPAs and the *provider* of a variety of telecommunications services to customers (EUAs). The SPA acts as a (currently very simple) *matchmaker*, finding suitable agents (NPAs), and accessing them on behalf of the requesting agent (EUA). In the future, this entity will be enhanced by introducing more sophisticated *brokering* and *recruiting* capabilities.

### 2.2 Communication Infrastructure

All NPI-mas agents communications rely upon the use of TCP/IP sockets. The first level of inter-operability is achieved through the use of standard interaction protocols, i.e., standard sequences of messages. FIPA[7] is a non-profit standardisation group that aims to promote inter-operability of emerging agent-based applications, service and equipment. Among others, FIPA specifies a set of standard communication facilities. Some standard FIPA protocols have been adopted and some new ones have been defined. Standard protocols enable a semantics interpretation of a messages' sequence in the context of a conversation, however this is not enough to guarantee full inter-operability.

Once messages are delivered and properly received, it is necessary to understand and interpret them. For this purpose, agents need to agree upon a common Agent Communication Language (ACL) and about a common way of representing common knowledge, i.e., the content language. The first version of NPI-mas makes use of the *Knowledge Query and Manipulation Language* [5] (KQML), since KQML facilities are provided within the JATLite platform. In order to increase the potential of the NPI-mas paradigm, we developed communication facilities for FIPA ACL to be used within JATLite [2]. FIPA ACL is in fact gaining more and more consensus in the agent community since it has a stronger semantic basis than KQML.

Concerning the knowledge representation, we have focused on Virtual Private Network (VPN) services and a content language that allows the representation of objects, propositions and actions has been developed: the VPN-cl. For portability and re-usability reasons, the syntax of the VPN-cl language has been defined in XML [3]. Agents deploying the VPN-cl are able to:

- Process information about objects such as services, physical connections, offers and Service Level Agreements (SLAs).

[2] More details can be found at http://liawww.epfl.ch/∼calisti/ACL-LITE/
[3] More details can be found at http://liawww.epfl.ch/∼calisti/ACL-LITE/VPN/

- Specify actions such as new service configuration, new connection activation, reservation of a connection, definition of a specific SLA, etc.
- Formulate propositions to motivate refusals or inform other agents about the results of an operation. Propositions are used for instance to inform that the end-to-end delay between two end-points is too high for a certain class of services that a given offer is too expensive that a service has been established, etc.

Finally, a common ontology has been defined and deployed. An *ontology* is an explicit specification of the structure of a certain domain. This includes a vocabulary for referring to the subject area, and a set of logical statements expressing the constraints existing in the domain and restricting the implementation of the vocabulary. The NPI-mas ontology provides a vocabulary for representing and communicating knowledge about VPN service provisioning. Furthermore, a set of relationships and properties that hold for the entities denoted by this vocabulary has been fixed.

### 2.3 The Network Provider Agents' Structure

Since the main goal of our simulator is to make use of algorithms and techniques designed for the inter-domain QoS-based routing, the focus is on the development of Network Provider Agents. The internal structure of a NPA is shown in

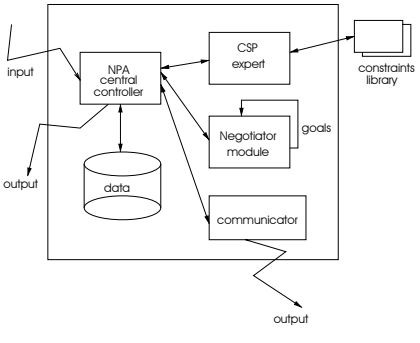


**Fig. 3.** *The internal NPA structure.*

Figure 3. The *perceptions* or *input* can be either messages coming from other agents or human user commands. The *central controller* is responsible for the coordination of several parallel activities, for processing inputs, for interfacing the agent world with human operators, for getting data characterising the network state. In our simulated scenario, the data is stored in a database that is dynamically updated during the simulation. In a real network, the data would be retrieved directly from the network management platform through the use of ad hoc wrappers. The NPA *communicator module* parses agents' messages, maintains communication channels, controls and manages all ongoing conversations.

The NPA *CSP expert* is a specialised sub-structure that is responsible for DCSP modelling and for applying typical DCSP consistency and searching techniques. The NPA *negotiator module* generates strategies for the controller. Considering the current state of the network, the various constraints, the utility function, and the interaction protocol, this module produces the strategy to be followed at every step of the negotiation process. Possible outputs of the NPA activity are either ACL messages to other agents, or internal actions, such as changes in the data configuration, or presentation of options to human operators.

### 2.4 NPA-to-NPA Interactions

Every NPA has an aggregated view of the multi-domain network topology that is deployed for computing the abstract paths to other networks in the environment. An *abstract path* is an ordered list of distinct providers' networks between the source and the destination nodes. The *initiator* is the NPA that first receives a service demand $d_k$ and computes one or several abstract paths along which the service negotiation is started. The choice of an abstract path $P$ is based on the following heuristics: (1) Eliminate all the paths that do not guarantee enough bandwidth. (2) Among the paths left select the cheapest (minimum cost). (3) If still more than one path exists, chose the path which has, after having accepted the incoming demand, the largest bandwidth left.

Next, the *initiator* contacts all the NPAs along $P$ requesting them to locally determine the set $S_k$ of possible internal routes for allocating $d_k$.

If all providers are locally consistent, i.e., $S_k$ is not empty, the *arc consistency* phase is started. The NPAs exchange information about inter-domain constraints. All incompatible network access points are discarded, so that every NPA reduces the set $S_k$. This phase involves a propagation of messages among neighbours, in order to revise the set $S_k$ for every access point that is discarded.

If arc consistency is successful (i.e, all NPAs have a non empty set $S_k$ consistent with inter-domain constraints) the negotiation for selecting a specific end-to-end route is started. The initiator broadcasts a *call-for-proposal* to all NPAs along $P$. A NPA's offer consists of a specific local route at a certain price. The *initiator* evaluates all received offers and elaborates possible global offers for the EUA, which involves pricing mechanisms and profit maximisation. The negotiation is successful whether the EUA accepts the offer. The initiator confirms to the NPAs the results of the transition. If the negotiation fails and the end-user does not modify its requirements the demand request is rejected and the initiator notifies all other NPAs.

## 3 The Lessons Learned

The availability of an increasing number of agent platforms, many of which are compliant to the FIPA standards [4], facilitates the work of developers of

[4] FIPA-OS is an open-source agent platform developed by Nortel Networks available at: http://www.nortelnetworks.com/fipa-os. JADE is an open-source software

agent-based applications that aim to be open and inter-operable with agent solutions implemented and managed by others. Nevertheless, even when deploying pre-existing tools, developers should be aware of a certain number of concrete technical issues that it is important to solve in an efficient way for not affecting the applications performance.

*Naming and addressing issues.* No matter what kind of platform is deployed, an agent needs to be uniquely identified. This obvious requirement, common to all distributed systems and usually solved by all the most common agent toolkits, can become a more complex issue when considering the need of supporting directory services, i.e., *yellow pages*. A FIPA compliant platform usually implements an Agent Communication Router (ACR) that is responsible for registering names and verifying their uniqueness. In addition, a *Directory Facilitator* (DF) entity records which services the registered agents can supply. In that case, the ability of an agent developer consists of creating standard, concise, but effective *identity cards* that the system, namely the DF, can flexibly and efficiently handle. Since JATLite does not supply any DF, an auxiliary entity devoted to 'yellow-paging' has been developed. The identity card of an NPI-mas agent consists of three major fields: the name, the profession and the address. The agent *name* is automatically checked by the NPI-mas Router whenever a new agent is created (the name must be unique). The *profession* identifies which kind of services the agent can support, namely if it is an EUA, a SPA or a NPA. Finally, the *address* includes the agent name, the host-name (i.e., the machine the agent is running in), and a time-stamp corresponding to the moment at which the agent has been created.

*Handling multiple conversations.* Every agent can be involved in parallel conversations. For this reason, an identifier, i.e., the *conv-id*, is used to identify a conversation for every non isolated communicative act. Therefore, vectors of *conversation* objects have to be dynamically maintained by every agent. At the same time, every agent needs to manage vectors of data structures, since different objects are instantiated for different and parallel allocation demands. The dynamic update of such vectors is the most delicate part of the data management, since several messages coming in parallel from different agents can concern the same data structure. To face this concurrency problem we adopted a FIFO policy, serialising the access to those data structures.

*Detection of a global state.* During the 'arc consistency' phase (see Section 2.4) there is a propagation of messages among agents in order to revise the set of all possible local routes $S_k$ for a demand $d_k$. This phase ends when all agents are consistent (or eventually as soon as one of them is inconsistent) and when no messages are pending in the system, i.e., when the *global state* is stable [3]. In order to detect that this state is reached a control mechanism, namely a Java thread instantiated by the NPA *initiator*, is used. The initiator receives notifications about the state of every single agent involved in the arc consistency and about the messages that are spread around in the system. This allows it to

framework that simplifies the implementation of multiagent systems developed by CSELT S.p.A. and available at: http://sharon.cselt.it/projects/jade

maintain and update a global state that is periodically checked by the control thread. However, this kind of mechanism has potential scalability problems due to the number of messages that are exchanged with the initiator NPA. For this reason, a pre-selection of intra-domain routes is important to reduce the number of updates in the set of possible local routes and therefore of messages exchanged in the system.

*Integration of JATLite with 'external' Java code.* The modularity of the JATLite's architecture enables developers to build agent-based systems that deploy only specific packages of such a tool. The on-line documentation of both JATLite and Java were sufficient for the integration of all the NPI-mas components. The development and the integration of new software components with the original JATLite structure enables FIPA ACL based communications and DF services. In addition, an auxiliary software package allows the usage of the VPN-cl language for facilitating agents negotiating about VPN service provisioning.

### 3.1 Evaluation of the NPI-Mas Paradigm

The performance metrics that have been observed for evaluating the NPI-mas paradigm and the DCSP-based techniques are the average demand allocation time, $T_{tot}$, and the allocation rate, $A_r := nbsuccess/nbdemands$, with $nbsuccess$ the number of demands successfully allocated, and $nbdemands$ the total number of service demands generated by the EUAs.

Simulations with a variable number $N_D$ of networks (Figure 4), and therefore of NPAs [5], have been run in order to check the scalability of our algorithms and in parallel the usability of communicating agents for the multi-provider service allocation purpose. The average values that has been obtained for $T_{tot}$, i.e.,

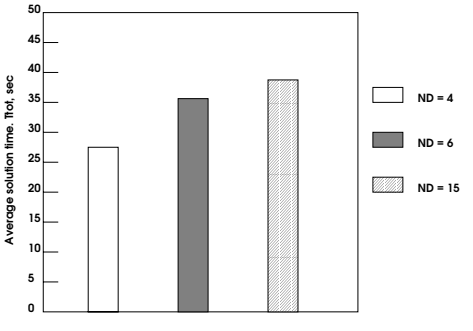


**Fig. 4.** *Three different simulated scenarios have been considered. $T_{tot}$ has been computed over a set of 300 randomly generated demands.*

$[25, 40]$ seconds, can be considered a positive result when compared to the current

[5] In our simulations $N_D \in [4, 15]$, which correspond to a realistic multi-provider environment. However, we are testing the NPI paradigm for greater values of $N_D$.

delays required by human operators to negotiate solutions and topology changes. Given a fixed number $N_D$ of providers' networks, we then tested the influence of a different number $|L|$ of inter-domain links. Increasing $|L|$ has a double effect: (1) The complexity of the solving process increases by augmenting the number of possible access points combinations between neighbour networks, therefore $T_{tot}$ increases, see Figure 5. However, the increment of $T_{tot}$ is smaller than 1/3, since the addition of new links does necessarily increases the search space for all service demand allocations. (2) The probability that at least one end-to-end path satisfying the QoS requirements of demands to be allocated exists augments so that $A_r$ increases, see Figure 6. Adding 1/3 of $|L|$ does not lead to an equivalent increment of $A_r$, since new links do not always facilitate the allocation for all possible service demands. Similar results have been obtained when varying the

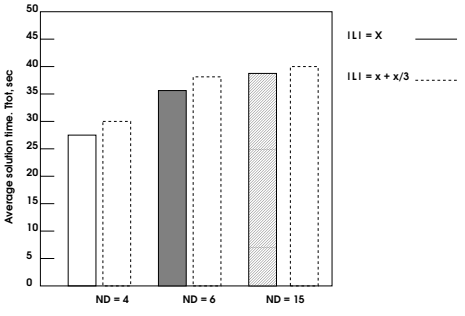


**Fig. 5.** *$T_{tot}$ increases when increasing the number of inter-domain links, since the search space become larger.*

complexity of the internal topology of every network. A greater number of network nodes and intra-domain links can augment the search space and introduce an additional computational overhead. However, similarly to the augmentation of $|L|$, if the topological changes correspond to an increment of available network resources, the number of demands successfully allocated increases.

### 3.2 Evaluation of the Simulator

The service demand allocation problem represents a complex and articulated process becomes even more complicated for networks that aim to provide QoS guarantees [4]. This is especially true in a multi-provider context where every provider tries to maximise his own utility and knowledge about the network topology is restricted by the network providers for strategic reasons. Although multi-domain QoS routing has been tackled from many viewpoints (ATM and SDH network [9], billing [1], multi-domain management in the MISA [6] project, agent interactions for routing multimedia traffic over distributed networks, see [6]

[6] http://www.misa.ch

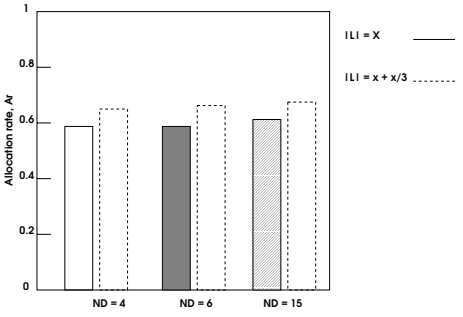


**Fig. 6.** *The graphic shows the increment of demands successfully allocated, when increasing the inter-domain capacity.*

and the FACTS project [7]) no previous work has addressed the possibility of dynamic negotiations about more than one path at a time.

Before starting the implementation of the NPI-mas system, an evaluation of different existing agent platforms were completed. Many multi-agent systems are available and the most common and traditional classifications distinguish between architectures for *reactive agents*, *deliberative agents* and *interacting agents* (see [10] and [13] for good overviews). Hybrid architectures have also been developed for integrating reaction and deliberation. However, some platforms are very specific to the traditional mentalistic approach, some are not freely available, and more in general no pre-existing agent platform was offering a support for any kind of CSP technique. Furthermore, when the NPI-mas project started, no FIPA compliant agent platform was publicly available. For these reasons, an ad hoc simulator has been developed from the scratch by deploying part of the JATLite architecture (the most general part) and adding new and more specific components.

In NPI-mas the use of Distributed Constraint Satisfaction techniques and the deployment of autonomous agents making use of a compact aggregated representation of network resources availability offer the capability of:

- Accelerating the allocation of multi-provider service demands, by automating many steps currently performed by humans.
- Supplying standard solutions that abstract from technical details, by using a common and standard agent communication language and a standard ontology. Standard agent communications, either via KQML or FIPA, facilitate the access to electronic service negotiations to a larger number of potential customers or sellers of Telecom services. The usage of agent communication languages instead of low levels languages such as SNMP (Simple Network Management Protocol) primitives or CMIP (Common Management Information Protocol) routines allows the abstraction from technical details and enhance agents' conversations by means of a semantics foundation behind the messages' sequence.

[7] http://www.labs.bt.com/profsoc/facts

- Supplying consistent solutions without the need for every participants to reveal internal and confidential data, such as network topologies, negotiation strategies, pricing mechanisms, etc.
- Supporting human decisions, or, in a more future scenario, of replacing human operators. Software entities can in fact accelerate the utility computation, optimise over different possible choices, follow more easily complex negotiation strategies, etc.
- Integrating economic principles within *self-interested* agents, in order to optimise the revenue.

NPI-mas agents could be integrated within a real network by providing ad hoc *wrappers* in order to interface the agent with the non-agent world, i.e., the network management and control level. These wrappers would take into account the low level characteristics of specific underlying network technologies. Although, it is not easy to perfectly map between the agents' understanding and the management's details, the abstraction provided by those wrappers would allows a more understandable and natural approach to many networking aspects otherwise very difficult to understand. The difficulty is related to the very low levels terms currently used for managing and controlling a network.

## 4 Conclusion

Currently many aspects of interworking are statically fixed using contracts and many steps of the interaction are carried out by human operators by fax, e-mail etc. The NPI-mas paradigm can be considered as a high level service which could be deployed in two different ways: in a short term period as a smart support for human operators, in a long term perspective as an autonomous system acting on behalf of humans and working at the connection management level. The first version would offer a valid support for human operators: it could compute local routes guaranteeing the QoS required and automatically verify which ones are not consistent with the inter-domain constraints (which are for instance the routes fixed in the currently used contracts). In a future scenario NPI-mas agents could supply an automated mechanism to route and negotiate the allocation of demands across distinct networks without the need for human intervention.

Implementing the NPI-mas system has been essential to validate theoretical concepts previously defined. The results obtained through the simulation prove the potential of our paradigm and estimate the performance.

Beyond more realistic simulations and more exhaustive data analysis, there are several directions that the authors are considering for the future development of NPI-mas.

- The introduction of more sophisticated negotiation techniques. This implies the definition of new protocols and/or new negotiation strategies. The gap between economic principles and software entities requires may force us to accept a "good" solution instead of looking for the optimal one.
- The development of alternative pricing mechanisms, such as auctions.

- The introduction of more complete and realistic representations of networking data and multi-provider service demands. For that purpose, more focus on current the service level agreements definitions is needed. Since there is no a standard and common way of expressing SLAs, it is non trivial to define a uniform representation that agents can use.
- The possibility of creating *coalitions* among NPAs, in order to take advantage of a higher degree of coordination.
- More work on the *service provider* level, i.e., more sophisticated brokering capabilities need to be added to the SPAs agents.

## References


1. C. Bleakley, W. Donnelly, A. Lindgren, and H. Vuorela. TMN specifications to support inter-domain exchange of accounting, billing and charging information. *Lecture Notes in Computer Science*, 1238, 1997.
2. M. Calisti and B. Faltings. A multi-agent paradigm for the Inter-domain Demand Allocation process. *DSOM'99, Tenth IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, 1999.
3. K. Mani Chandy and Leslie Lamport. Distributed snapshots: Determining global states of distributed systems. *TOCS*, 3(1):63–75, February 1985.
4. Shigang Chen and Klara Nahrstedt. An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions. *IEEE Network*, pages 64–79, November/December 1998.
5. Tim Finin et al. Specification of the KQML Agent-Communication Language – plus example agent policies and architectures, 1993.
6. Fipa. Fipa spec. 97 v2.0. *Foundation for Intelligent Physical Agents*, 1997.
7. Foundation for Intelligent Physical Agents. FIPA Specifications, October 1997. http://www.fipa.org/spec/.
8. Alex L. G. Hayzelden and John Bigham, editors. *Software Agents for Future Communication Systems: Agent Based Digital Communication*. Springer-Verlag, Berlin Germany, 1999.
9. D. Karali, F. Karayannis, K. Berdekas, and J. Reilly. Qos based multi-domain routing in public broadband networks. *Lecture Notes in Computer Science*, 1430, 1998.
10. J. P. Muller. The design of intelligent agents: a layered approach. *Lecture Notes in Artificial Intelligence and Lecture Notes in Computer Science*, 1177, 1996.
11. Edward Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London, UK, 1993.
12. S. N. Willmott and B. Faltings. Active Organisations for Routing. In S. Covaci, editor, *Proceedings of the First International Working Conference on Active Networks.*, pages 262–273. Springer Verlag, Lecture Notes in Computer Science Series, number 1653, 1999.
13. M. Wooldridge and N.R. Jennings. Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.
14. M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. Distributed Constraint Satisfaction for Formalising Distributed Problem Solving. *Proceedings 12th IEEE International Conference on Distributed Computing Systems.*, pages 614–621, 1992.