A distributed approach for QoS-based multi-domain routing

M. Calisti, C. Frei and B. Faltings
Laboratoire d'Intelligence Artificielle
Swiss Federal Institute of Technology
CH-1015 Lausanne, Switzerland
{calisti, frei, faltings}@lia.di.epfl.ch

Abstract

Today's networks are controlled at various organisational and functional layers by human managers. What seems more suitable for the future scenarios is a management solution based on static and/or mobile software entities, collecting network state information and having the ability to directly invoke effective changes to switch controllers, without the interaction of a human operator. In particular, this would enable more flexible interactions among distinct network providers improving the process of allocating a connection which spans several networks. This paper describes a software architecture which makes use of Distributed Artificial Intelligent techniques in order to support the QoS-based multi-domain routing process.

Introduction

In the deregulated telecommunications market improving interworking is very important since it is a prerequisite for supporting advanced services spanning several domains. This task is even more delicate for networks which aim to provide any kind of Quality of Service (QoS) guarantees.

Currently many aspects of the interworking are statically fixed by contracts (number and available capacity of links connecting one network domain to another, prices etc.) and many steps of the interaction are regulated by human operators via fax, e-mail, etc. This makes the overall inter-operability process very slow (for instance several months can pass before effective inter-domain network configuration changes take place) and quite inefficient. Inefficiency is mainly due to the difficulty for human operators to consider all aspects which complicate the interworking process:

- The increasing number of actors (content providers, added-value service provider, brokers,etc.) with different roles than those of the traditional telecom operators.
- The use of different and heterogeneous technologies.
- The need for a mapping between intra- and inter-domain management aspects.

Among all various interworking aspects, the end-to-end routing for networks which support QoS guarantees is a very complex problem (Chen & Nahrstedt 1998; Lin & Yee 1989; Lee 1995). Routing decisions can become very complex (often NP complete to route just a single demand) whenever various QoS parameters such as bandwidth, delay, jitter, cell-loss ratio, etc. must be taken into account.

The complexity increases whenever the end-points belong to network domains owned by distinct authorities, i.e., in a *multi-domain scenario.* In this latter case, the routing task is made very difficult by the fact that individual network providers do not reveal detailed information about their internal network.

This paper focuses on the QoS-based multi-domain routing process and introduces a distributed paradigm that enables routing decisions making use of restricted information. In particular, it is described how Artificial Intelligence techniques for distributed problem solving supply a compact way to formalise the *inter-domain routing process*, i.e., routing between provider domains, and how this formalism enables an agent middleware to actively route demands. Although, the *intra-domain* process, i.e., the routing within a single network domain, is not considered in this paper, the technique we propose shows how the intra-domain process can be made consistent with the inter-domain routing.

Global framework

The network of a provider A *provider's network*, or network, is modelled as a graph, as depicted in Figure 1. A node corresponds to a network node (such as switches and routers) or even to sub-networks. The set of links interconnecting the nodes are *intra-domain links*, and represent the communication channels existing inside every network. Communication between the networks of the providers take place through *inter-domain links* (see Figure 1).

Every link l_i, either intra- or inter-domain, is characterised by:

- A vector qos_i expressing the Quality of Service properties of the link: available bandwidth (typically measured in [bits/second]), and other parameters such as delay ([μs]), bit loss ratio, bit error rate, etc.

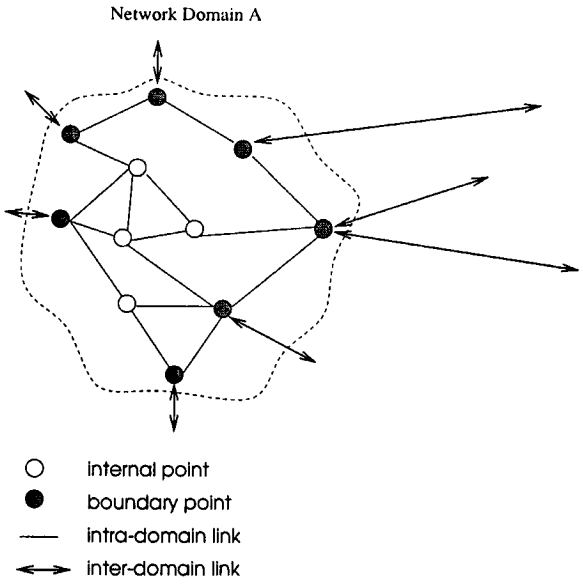

Figure 1: Network of a provider and the inter-domain links it is connected to.

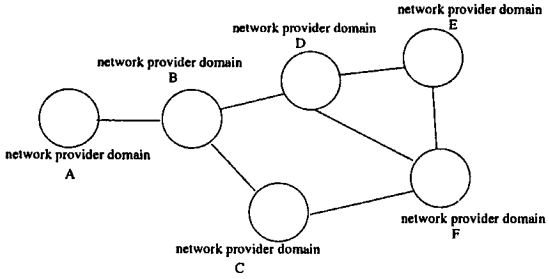

Figure 2: Providers' graph.

- The cost c_i of the link (also called price), which is assumed to be fix (and known) for the overall duration of the inter-domain routing process.

The providers' graph The interconnection of the different providers' networks can be summarised in an abstract simple graph, called the *providers' graph*, as shown in Figure 2. The providers' graph consists of abstract nodes and abstract links. An *abstract node* represents a network provider domain, and is characterised by a *node traversal delay* and a *node traversal cost*. An *abstract link* between two abstract nodes clusters all inter-domain links interconnecting the two corresponding network provider domains.

The demands In this framework, a communication demand d_k, or *demand*, is specified by a triple:

$$d_k ::= (x_k, y_k, qos_{req,k})$$

where x_k is the source node, y_k the destination node, and $qos_{req,k}$ the required QoS by the demand. A demand may be anything from a phone call to a virtual link in a Virtual Private Network.

$qos_{req,k}$ is restricted to bandwidth considerations within the scope of this paper. Therefore, $qos_{req,k}$ expresses the maximal bandwidth (peak bandwidth) needed by the demand. This includes all demands with Constant Bit Rate at a *constant rate* $\leq qos_{req,k}$, or Variable Bit Rate with *peak rate* $\leq qos_{req,k}$.

Problem definition

The process of allocating a route for a service demand which spans several providers' networks is a very complex task for several reasons: (1) there are distinct entities involved (final customers, service providers, etc.), (2) the routing must take into account QoS requirements, (3) network resources and information are distributed, (4) a trade-off between the profit optimisation and the end-user (i.e., the service customer) satisfaction is required.

Two main sub-problems need to be addressed: (1) Finding the routes that satisfy connectivity and QoS constraints. (2) Selecting a specific route by negotiating with other providers. This paper focuses on the former sub-task that can be more precisely expressed as it follows:

When a network provider receives a demand, it has:

- To detect the source and the destination network domains. Whenever the destination node resides in a remote network provider domain the *inter-domain routing* process must be started[1].
- To compute an abstract path P. An *abstract path* is an ordered list of distinct network provider domains between the source and the destination network provider domains.
- To contact all the network providers along P.
- To compute the *local routes*, i.e., intra-domain routing inside the network providers along P.
- To make the set of local routes consistent with inter-domain constraints. All local routes which violates such constraints are discarded.
- To negotiate with the providers along P in order to allocate a global route. A *global route* is the end-to-end connection consisting of local routes and inter-domain links interconnecting them.
  - If an agreement is found, the network resources are reserved,
  - Otherwise the service demand is rejected.

The Network Provider Interworking paradigm

Every network provider is represented by a *Network Provider Agent*, NPA (for the large amount of goals and

[1] If the source and the destination nodes belong to the same network provider domain the *intra-domain routing* is started.

tasks, this entity is implemented by means of several agents, i.e., negotiator, contractor etc.).

The inter-domain routing requires the coordination of distributed NPAs, since all knowledge about the resources needed to allocate a demand cannot be gathered into one agent, i.e., into one single network provider, for various reasons. First of all, collecting information about a problem implies communication costs and it can become very inefficient or even impossible for scalability reasons. Acquiring and maintaining data about connectivity and QoS aspects of the overall scenario it is practically infeasible due to the large number of network providers and the complexity of every network domain. Finally, collecting all the information into one agent may not be desirable because of security and privacy policies.

In our framework, distributed agents solve the inter-domain routing problem without centralising all information: every NPA assigns a part of the global route, namely the local part inside its network, and it negotiates with others in order to interconnect its local route to form a global end-to-end connection. However, a minimal amount of information about the external networks is needed. Every NPA needs in fact to have a global view, i.e., the *providers' graph*, in order to be able to compute abstract paths.

At this stage of the development the NPI paradigm does not depend on the network technology which provides QoS guarantees, it could be either ATM or Internet[2] supporting per flow guarantees.

Formalism for the multi-domain routing process

Constraint satisfaction is a powerful and extensively used Artificial Intelligence paradigm (Tsang 1993). *Constraint satisfaction problems* (CSP) involve finding values for problem variables subject to restrictions (constraints) on which combinations of values are acceptable. CSP are solved using search (e.g., backtrack) and inference (e.g., arc consistency) methods. A *Distributed Constraint Satisfaction Problem* (DCSP) (Yokoo *et al.* 1992) is a CSP that is solved by distributed agents, each agent being responsible for a set of variables, and exchanging messages with other agents to prevent the violation of any constraint.

Finding a route for allocating service demands that cross distinct networks can be considered as a DCSP, since the variables are distributed among agents and since constraints exist among them. We assume that: (1) every agent has exactly one variable, (2) all inter-domain constraints are binary, i.e., they involve two variables, (3) intra-domain constraints are unary and must ensure that at least one local path verifies the QoS requirements, (4) there is at least an agent for

[2]IP with the advent of flow identification (IPv6) and per flow routing or per application (that may be possible with active network technology) offer differentiated levels of QoS.

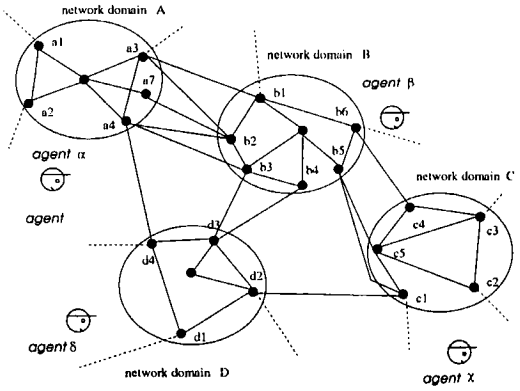

Figure 3: Network representation.

every domain, (5) all the agents in the scenario know each other, (6) agents communicate using messages.

The DCSP can be represented as a graph, the *constraint graph*, where variables are vertices and constraints are edges between vertices. Note that this constraint graph is not the physical communication network. An edge in the constraint graph is not a physical communication link, but a logical relation between agents. Since each agent owns exactly one variable, a vertex in the constraint graph also represents an agent.

The *variable* every agent handles is a "local path" (actually it is not a path since it expresses just the two end-points of the local path) specified as a couple:

$$p ::= (inputpoint, outputpoint)$$

The values for each "local path" are all the possible combinations of boundary points i.e., input-output points, which represent the possible local routes to allocate to the demand. The *boundary points* are nodes which connect every network provider domain to external networks. Note that only *simple paths*, i.e., loop-free, are considered.

The set of all possible input-output points combinations is the *domain* for each variable.

Consider the example depicted in Figure 3. Agent α receives a demand $d_k = (a_1, b_6, qos_{req})$, and, for instance, it selects the abstract path A - B. Next, agent α determines:

$$I_{\alpha,k} ::= \{ \text{ Set of possible input points } \}$$
$$O_{\alpha,k} ::= \{ \text{ Set of possible output points } \}$$

In this example $I_{\alpha,k} = \{a_1\}$ and $O_{\alpha,k} = \{a_3, a_4, a_7\}$, since the only output points directly connected to B are a_3, a_4 and a_7. The variable for agent α is the couple $p_\alpha = (i, o)$, $i \in I_{\alpha,k}$ and $o \in O_{\alpha,k}$. The *domain* for p_α is given by the set of all the possible routes connecting i to o: $D_{\alpha,k}$= $\{(a_1, a_3), (a_1, a_4), (a_1, a_7)\}$. Agent β owns the variable $p_\beta = (i, o)$, with $i \in I_{\beta,k} = \{b_1, b_2, b_3\}$ and $o \in O_{\alpha,k} = \{b_6\}$. Considering only the points, which are directly connected with the predecessor or with the successor along the path, allows to reduce the complexity of the solving algorithm. The

fewer points there are in $I_{\alpha,k}$ and $O_{\alpha,k}$, the fewer possible combinations there are for allocating a demand (search space reduction). The domain for agent β is: $D_{\beta,k} = \{(b_1, b_6), (b_2, b_6), (b_3, b_6)\}$.

The domains of the variables are dynamically calculated for every specific demand. The dynamical recomputation allows: (1) to update the variable domains according to the network state, (2) to reduce the search space.

The constraints

Connectivity and QoS constraints are locally translated in constraints between the boundary points that two neighbour network domains use for the inter-domain routing. First of all it is necessary to check which are the possible combinations of input/output points from/to each other neighbour network. The set S is defined:

$$S(\alpha,\beta) ::= \{(o_\alpha, i_\beta) \mid o_\alpha \in O_{\alpha,k}, i_\beta \in I_{\beta,k}, o_\alpha \rightsquigarrow i_\beta\}$$

$o_\alpha \rightsquigarrow i_\beta$ means that o_α is directly connected to i_β.

For N_α nodes in the network A and N_β in the domain B there are at most $N_\alpha \times N_\beta$ possible inter-domain connections, i.e., number of elements in S.

Considering Figure 3, the set S of possible combinations of output points of network A and the input points of network B is: $S(\alpha,\beta) = \{(a_3, b_1), (a_3, b_2), (a_4, b_3), (a_4, b_2), (a_7, b_2)\}$.

We can formalise and express all the possible constraints between two agents α and β as follows:

$$C(\alpha,\beta) ::= \{\ ((i_\alpha, o_\alpha), (i_\beta, o_\beta)) \mid (o_\alpha, i_\beta) \in S(\alpha,\beta), (i_\alpha, o_\alpha) \in D_\alpha, (i_\beta, o_\beta) \in D_\beta\ \}$$

Note that both $S(\alpha,\beta)$ and $C(\alpha,\beta)$ are dynamically calculated for every specific demand.

Distributed Arc Consistency - based solving technique

The *Distributed Arc Consistency* (DAC) algorithm is based on the use of the arc consistency technique (Kumar 1992) applied to a very simple case of constraint graph. In our framework, once an abstract path P has been selected, the constraint graph is a simple chain. Arc consistency is a technique used to narrow the space of possible choices before actually performing search: every link $i - j$ of the constraint network is made *arc consistent* by eliminating all the values of the variables i and j which are not consistent with the given constraint.

Every NPA along P tries to define the *feasible set* of possible local paths satisfying the QoS requirements of the demand to be allocated. Then a filtering function (AC propagation) eliminates inconsistent values from the feasible set. The feasible set finally contains the set of consistent values for the agent's variable.

DAC consists of the following main steps:

1. An agent α receives a service demand d.
2. Agent α detects the destination and the source network domains.
3. Based on the providers' graph, which is dynamically updated by the agents, an abstract path P for d is computed. None, one or several paths may exist. If none, the demand is rejected. If more than one path exists, the shortest one which satisfies QoS constraints is selected.
4. For the selected path agent α contacts every agent along P. From now on, several agents run similar routines in parallel.
5. Each agent defines the variable domain D and the set of constraints C. C is determined considering the available QoS on the links l_i which interconnect every provider network with its neighbours.
6. Based on qos_{req} every agent reduces its variable domain D by performing *node consistency*: every local path (i_α, o_α) that cannot support the required QoS is eliminated from the domain D. If an agent has an empty variable domain D a failure message is sent to all agents along P. A new abstract path must be investigated (go to 3).
7. Establishing arc consistency: every agent determines all the values of its domain which are compatible with the values contained in the domains of the neighbours. If an agent obtains an empty variable domain D a failure message is sent to all agents along P: d cannot be allocated along P. Go to 3.
8. At least one solution, i.e., one route along P, exists. In order to be sure that a solution will still exist after the negotiation, we assume that there exists a pre-reservation mechanism of the negotiated resources required by the demand. If network failures or physical changes occur the guarantee of having a solution is not valid any more. In this case every agent is notified by the Network Management System and the algorithm backtracks to step 3.
9. If the negotiation is successful the resources needed are reserved[3] and the service demand can be allocated, otherwise a failure message is sent to all the agents involved along P and the algorithm goes to 3.

The node consistency process (step 6) can be supported by the Blocking Island paradigm introduced in (Frei & Faltings 1997). A *blocking island* (BI) is resource abstraction technique that allows to quickly assess the existence of routes between end-points with a given amount of available bandwidth, without having to explicitly search for such a route. More precisely, if the input point i_α and the end point o_α belong to the same BI at the required QoS level, then $(i_\alpha, o_\alpha) \in D$, i.e., there is a local route which guarantees the qos_{req}. Note that the choice of internal routes (in step 8), when

[3]Note that you pass from a pre-reservation, step 8, which is temporary and for all the potential resources involved, to an effective reservation of just those resources that are effectively required for the specific demand.

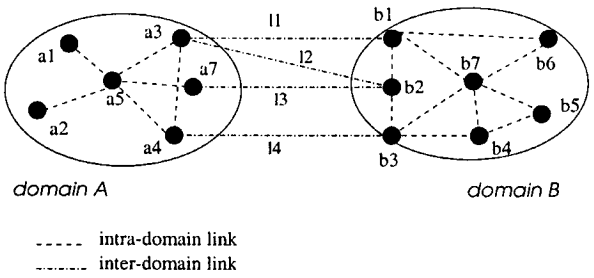

Figure 4: Simple global scenario.

there are more than one, can rely on BI-based route selection heuristics, as shown in (Frei & Faltings 1997), in order to achieve load-balancing within a provider's network.

Visualising the main steps of DAC The following examples graphically show the main steps of the DAC algorithm. The DCSP formalism allows to easily visualise which are the links in the scenario that guarantee the QoS required. After the arc consistency propagation the only links considered are the ones which guarantee qos_{req} and which are consistent, i.e., which satisfy the inter-domain constraints. Consider that NPA A receives the demand request $d = (a1, b6, qos_{req})$. The scenario is showed in Figure 4.

Figure 5 (A) shows the situation before any computation. The routes inside every domain are specified as couples of end-points, e.g., $(a1, a3)$, inside the network domain A.

Figure 5 (B) indicates all the inter-domain links and local routes that satisfy the QoS requirements. This is the configuration after every agent has performed steps 5 and 6 of the DAC algorithm. At step 5 the links $l2$ and $l3$ are excluded since they cannot guarantee the qos required. Step 6: the local path $(a1, a7)$ is pruned out from the variable domain D_α and the local paths $(b2, b6)$ and $(b3, b6)$ are pruned out from the variable domain D_β. Finally, (B) shows $D_\alpha = \{(a1, a3), (a1, a4)\}$ and $D_\beta = \{(b1, b6)\}$.

Figure 5 (C) depicts the situation after arc consistency propagation (step 7 of the DAC algorithm). The variable domains are: $D_\alpha = \{(a1, a3)\}$ and $D_\beta = \{(b1, b6)\}$. The value $(a1, a4)$ is pruned out from D_α since $(a1, a4)$ is not consistent with any value in D_β.

There is in the end one solution given by: $(a1, a3), l1, (b1, b6)$. The agents can still negotiate about prices. Different internal routes can have in fact different costs. Figure 5 (D) shows one specific global route.

Efficiency of the AC approach

The arc consistency approach is particularly suitable first because it guarantees the completeness of our algorithm, and second because of the simple constraint graph (once an abstract path is selected the constraint graph becomes a simple chain).

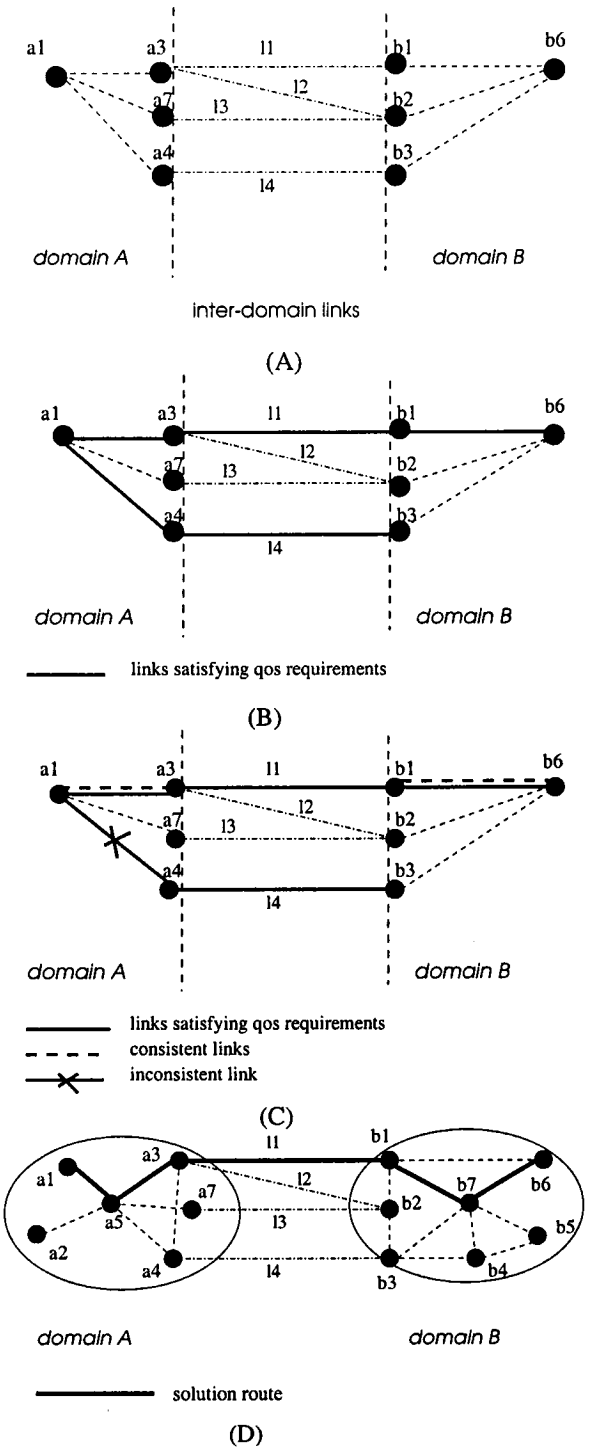

Figure 5: Example 1: at the end of the arc consistency propagation there is just one possible solution.

Performing arc consistency for a binary CSP is bound by $O(ed^3)$ (Mackworth 1977), where e is the number of constraints, and d the upper bound on the number of values in the domain of a variable.

Let P be the abstract path chosen for the current demand. In our problem, the number of binary constraints is $\mid P \mid -1$, that is the number of variables (or involved domains/agents) minus 1.

In order to compute the domain size $\mid D_A \mid$ of a variable A, we must consider all border nodes of a provider's network. Let n_A be that number. There are two exclusive cases (remember that we suppose that source and destination end-points are not in the same domain – otherwise no inter-domain routing is required):

1. The network domain contains the source node or the destination node, but not both: $\mid D_A \mid \leq n_A$.
2. The network domain is a transit domain, i.e., it does not contain either the source or the destination node, but is part of the abstract path P: $\mid D_A \mid \leq \frac{n_A(n_A-1)}{2}$.

Let N be the maximal amount of boundary nodes in the network of a provider, i.e., $N = \max_{\text{domains A}} n_A$. Therefore, the complexity of the arc consistency process is bound by $O(\mid P \mid N^6)$.

The following theorem proves that the DAC algorithm does not require any search (Freuder 1988; 1984):

Theorem 1 (Freuder82) *If a constraint graph is strongly k-consistent, and $k > \omega$, where ω is the width of the constraint graph, then there exists a search order that is backtrack free.*

In our case, the constraint graph has a width $\omega = 1$, since once an abstract path has been selected, the constraint graph is a simple chain[4]. The DAC algorithm uses node consistency and arc consistency to make the graph strongly 2-consistent. Therefore, our CSP can be solved without any search after performing node and arc consistency.

Discussion

The NPI paradigm can be considered as an active 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. In the short term period the algorithm proposed could dynamically supply a set of possible solutions for the inter-domain routing, could compute the intra-domain routes which can support the required QoS and could automatically verify which local routes are not consistent with the inter-domain constraints (which are for instance the ones fixed in the currently used contracts). In a future scenario NPI could supply an automated mechanism to route and negotiate the allocation of demands across distinct domains without the need for human intervention. This would require further work to integrate NPI in a real network infrastructure (for instance telecommunications architectures such as TINA, IN, or more generically TMN (Hall 1996) compliant environments).

[4]At least one of the orderings of a tree-structured constraint graph has a width equal to 1 (Freuder 1988; 1984).

Scenario	APC-time	T-time
4-ND 5-ID	18 msec	146 msec
4-ND 11-ID	50 msec	569 msec
8-ND 8-ID	23 msec	483 msec
8-ND 16-ID	138 msec	507 msec

Table 1: DAC time performance.

Some preliminary results for the first kind of deployment are given in Table 1. Although these values are strictly dependent on the simulated scenarios (i.e., Network Domains, ND, and Inter-Domain links, ID, configuration) and on the specific demand to be allocated, they give an estimation of the time needed by DAC to compute abstract routes (Abstract Path Computation time) and to find a solution (if a solution exists or to determine that no solution is available). Analogue results are summarised in Figure 6: in less than one second DAC gives a final answer to a specific network provider, provided a global view of the scenario. These preliminary results suffer from a lack of more realistic case studies.

Finally, it is important to underline that the proposed agent system assumes that agents inter-operate and negotiation criteria derive from the same ontology. Regarding the inter-operability among agents, NPI aims to be FIPA compliant (Foundation for Intelligent Physical Agents 199798). Agents can refer a common negotiation ontology and adopt a negotiation protocol which they commonly agree on (e.g., **fipa-contract-net**). We believe that the multi-agent technology can become a strategic instrument to improve many networking and interworking aspects only if standard interfaces guarantee communication and interaction between different agent applications, which will run at different levels in and above the network.

Conclusion

This paper has described a multi-agent paradigm to support the QoS-based inter-domain routing problem in a flexible and dynamic way without the need of human intervention. In particular, the paper shows how the DCSP formalism provides a powerful and intuitive way of expressing the QoS-based inter-domain routing problem, and how to solve it efficiently without brute-force search.

In our view, the increased flexibility of the NPI approach provides for new opportunities in the area of network inter-operability. In the short term such a scenario could be introduced through software tools supporting human operators, for example proposing alternative choices, evaluating prices, QoS and topology constraints, and summarising offers from other operators.

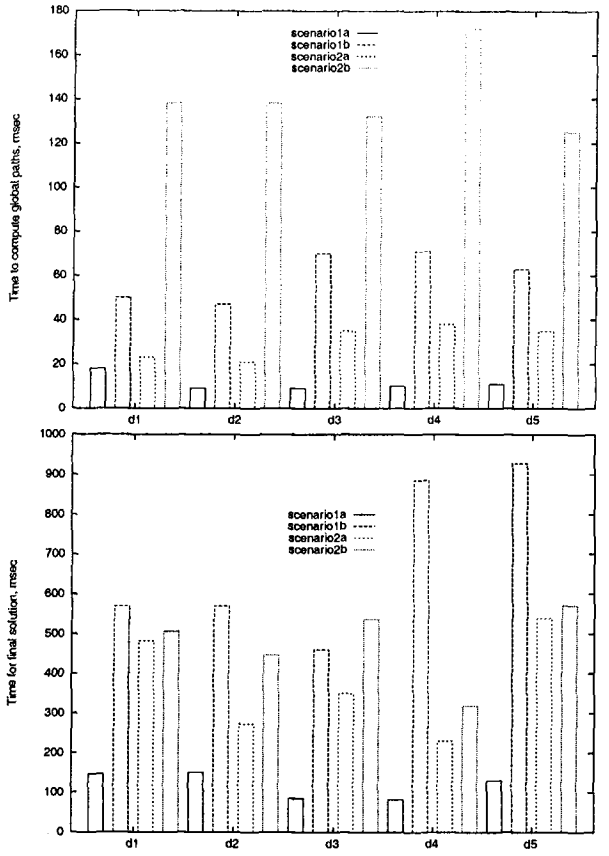

Figure 6: Time performance comparison for distinct demands.

In a future scenario, which could possibly be enhanced by active networking, NPI agents could program the behaviour of network nodes (switches and/or routers) in order to automate the inter-domain routing process. Furthermore, agents could supply a flexible and dynamic negotiation framework.

We have built prototypes to test many of the concepts outlined in this paper. In order to validate the NPI paradigm we are continuing to test it by constructing more realistic scenarios and designing negotiation protocols around them, a problem that is complicated by the fact that the interworking process is itself influx and no stable data is available.

References

Chen, S., and Nahrstedt, K. 1998. An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions. *IEEE Network* 64–79.

Foundation for Intelligent Physical Agents. 1997/98. FIPA Specifications. http://www.fipa.org/spec/.

Frei, C., and Faltings, B. 1997. A dynamic hierarchy of intelligent agents for network management. *Workshop on Artificial Intelligence in Distributed Information Networks IJCAI'97.*

Freuder, E. C. 1984. Direct independence of variables in constraint satisfaction problems. Technical Report 84-15, University of New Hampshire, Department of Computer Science.

Freuder, E. 1988. Backtrack-free and backtrack-bounded search. In Kanal, L., and Kumar, V., eds., *Search in Artificial Intelligence.* New York: Springer-Verlag.

Hall, J. 1996. Management of telecommunication systems and services: Modelling and implementing TMN-Based multi-domain management. *Lecture Notes in Computer Science* 1116.

Kumar, V. 1992. Algorithms for Constraint Satisfaction Problems: A Survey. *AI Magazine* 13(1):32–44.

Lee, W. C. 1995. Topology Aggregation for Hierachical Routing in ATM Networks. *ACM SIGCOMM* 82–92.

Lin, F. Y. S., and Yee, J, R. 1989. A Distributed Routing Algorithm for Virtual Circuit Data Networks. In *Proceedings IEEE INFOCOM.* 200–207.

Mackworth, A. K. 1977. Consistency in Networks of Relations. *Artificial Intelligence* 8:99–118.

Tsang, E. 1993. *Foundations of Constraint Satisfaction.* London, UK: Academic Press.

Yokoo, M.; Durfee, E. H.; Ishida, T.; and Kuwabara, K. 1992. Distributed Constraint Satisfaction for Formalising Distributed Problem Solving. *Proceedings 12th IEEE International Conference on Distributed Computing Systems.* 614–621.