Automated Negotiation for Provisioning Virtual Private Networks using FIPA-Compliant Agents

P. Faratin[1], N. R. Jennings[2], P. Buckle[3] and C. Sierra[4]

[1]Dept. of Electronic Engineering, Queen Mary and Westfield College, University of London, London E1 4NS, UK. Email: P.Faratin@qmw.ac.uk

[2] Dept. of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, UK. Email: nrj@ecs.soton.ac.uk

[3]Advanced IP Services and Management, Nortel Networks, Harlow Labs, Harlow, UK. Email: pbuckle@nortelnetworks.com

[4]IIIA- Artificial Intelligence Research Institute, CSIC-Spanish Council for Scientific Research, 08193 Bellaterra, Catalonia, Spain. Email: sierra@iiia.csic.es

ABSTRACT

This paper describes the design and implementation of negotiating agents for the task of provisioning virtual private networks. The agents and their interactions comply with the FIPA specification and they are implemented using the FIPA-OS agent framework. Particular attention is focused on the design and implementation of the negotiation algorithms.

1. INTRODUCTION

Multi-agent system (MAS) research focuses on how a society of autonomous, computational agents can interact in order to solve complex, real world problems that are inherently distributed in nature (Bond and Gasser, 1998). Due to the distributed and inter-related nature of the problem, one of the key issues in any MAS is that of coordination. To achieve coordination in a given application, a number of sub-problems need to be addressed (Weiss, 1999). Firstly, there is a need to define a *language* and a *protocol* of interaction. The former specifies the syntax and semantics of the communication. The latter specifies the normative (or constraining) rules of who can say what in the course of the interaction. Secondly, there is a need to specify how agents will reason about the coordination problem. That is, there is a need to model how agents will represent and reason about the actions, plans and knowledge of themselves and others. Thirdly, there is a need to identify how coherent system-wide behaviour can be attained from agents that invariably have a local and partial view of the problem. Finally, in addition to the aforementioned modelling problems, there is also a need to actually *engineer* a practical MAS through the design of platforms and methodologies (Bond and Gasser, 1998).

The work reported here addresses all the aforementioned coordination sub-problems. The engineering aspects are addressed by adopting the domain-independent software standard formulated by the Foundation for Intelligent Physical Agents (FIPA). FIPA specifies the normative standard for the syntax and semantics of the language used during agent interaction, as well as the protocol of interaction. Agents are then embedded within this environment and interact, complying with the standard, to solve their domain problem. The problem domain we consider is the competitive and dynamic provisioning of virtual private networks (VPNs) by end

users. Agents in this scenario reason about the coordination process by using a novel negotiation model that we have developed and a FIPA-compliant negotiation protocol. This model aids the agents in reaching agreements and in coordinating their interactions, while respecting their computational and information boundedness.

The contribution of this work is two fold. Firstly, we present a new agent platform (called **FIPA-OS**) that is both practical and generic. This is the first, freely available implementation of the FIPA standard. Secondly, we present the design and utilisation of a negotiation model that is suitable for a range of real-world applications. This model advances the state of the art in that it is based upon practical assumptions and that it incorporates a wide range of negotiation techniques (including concession making, making trade-offs between issues and issue modification during the course of an ongoing negotiation).

The remainder of the paper is structured as follows. Section 2 presents the FIPA-OS platform. Section 3 introduces the VPN scenario and section 4 describes its implementation using FIPA-OS. This scenario is then used to motivate and detail the negotiation model in section 5. Finally, section 6 presents the conclusions and future work

2. FIPA-OS

The purpose of FIPA is to promote the development of specifications of generic agent technologies that maximise interoperability within and across agent based applications. Part of its function is to produce a specification for an agent enabling software framework. Contributors are free to produce their own implementations of this software framework as long as its construction and operation complies with the published FIPA specification. In this way the individual software frameworks are interoperable. Within this context, FIPA-OS[1] is an open source implementation of the mandatory elements contained within the FIPA specification for agent interoperability. The FIPA-OS distribution contains class files, Java source code and documentation. It also includes a simple test agent to access the agent platform services and some visualisation software.

The FIPA-OS architecture can be envisaged as a non- strict layered model. In a non-strict layered model, entities in non-adjacent layers can access each other directly. The developer is able to extend the architecture, not only by appending value-added layers (such as specialist service agents or facilitator agents on top) but in addition, lower or mid layers can be replaced, modified or deleted. In addition to the mandatory components of the FIPA Reference Model (see section 2.1), the FIPA-OS distribution includes support for:

- Different types of Agent Shells for producing agents which can then communicate with each other using the FIPA-OS facilities;
- Multi-layered support for agent communication;
- Message and conversation management;
- Dynamic platform configuration to support multiple communication infrastructures and multiple types of persistence (enabling integration with legacy persistence software);
- Abstract interfaces and software design patterns.

[1] FIPA-OS is an agent framework, originating from research at Nortel Networks' Harlow Laboratories in the UK. It is available as managed Open Source [FIPA-OS URL] and is currently used within a number of industrial and academic institutions, including the ACTS project FACTS [FACTS URL].

2.1 FIPA Reference Model

The FIPA reference model illustrates the core components of the FIPA-OS distribution (figure 1). The agent reference model provides the normative framework within which FIPA agents exist and operate. Combined with the FIPA agent life cycle, it establishes the logical and temporal contexts for the creation, operation and retirement of agents.

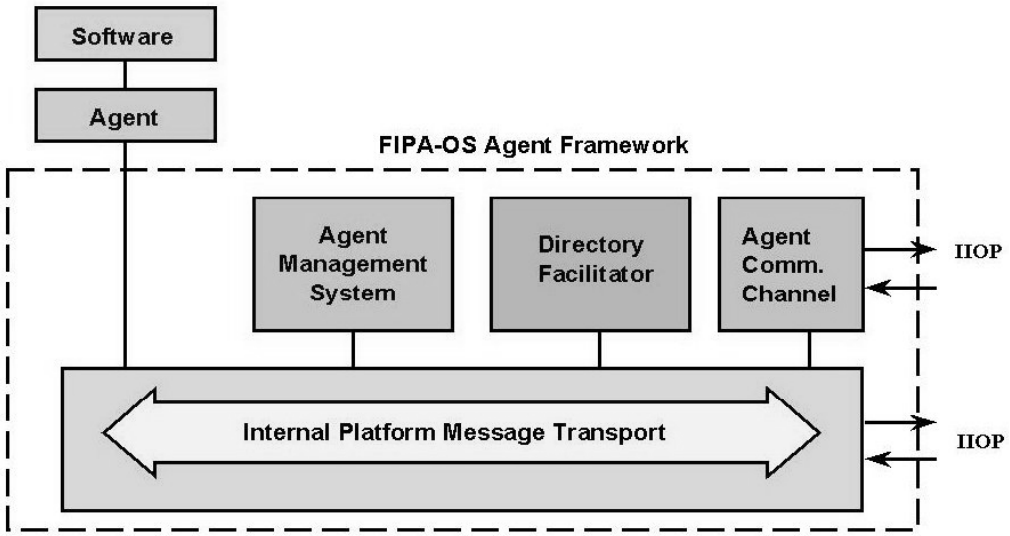

Figure 1: FIPA-OS Agent Framework

The Directory Facilitator (DF), Agent Management System (AMS) and Agent Communication Channel (ACC) are specific types of agents, which support agent management. The DF provides "yellow pages" services to other agents. The AMS and ACC support inter-agent communication using FIPA's agent communication language (ACL). The ACC supports interoperability both within and across different platforms. The Internal Platform Message Transport (IPMT) provides a message routing service for agents on a particular platform that must be reliable, orderly and adhere to the detailed requirements specified in section 5.2 of (FIPA 97 V2, Part 2). Together, the ACC, AMS, IPMT and DF form what will be termed the *Agent Platform* (AP). These are mandatory, normative components of the model. For further information on the FIPA Agent Platform see (FIPA 97 V2, Part 1, Agent Management) and (FIPA 98, Part 13).

In addition to the mandatory components of the FIPA Reference Model, the FIPA-OS distribution includes an Agent Shell that can be seen as an empty template for an agent. Multiple agents that can communicate with each other using the FIPA-OS facilities can then be produced from this template. Details of how this Agent Shell has been used to create the agents described in this paper are given in section 3.1.

3. Designing VPN Provisioning Agents

The scenario we consider is the provisioning of a public communication network (such as the Internet) as a virtual private network (VPN) for end users. In the real world, a single service to an end user consists of different combinations of a range of network services (see figure 2). These network services can be broadly decomposed into three domains, which may be represented by agents. The three domains and their representative agents are:

1. End User Communication Domain: *Personal Communication Agent* (PCA).
2. Network Services Domain: *Service Provider Domain Agent* (SPA).
3. Network Connection Domain: *Network Provider Domain Agent* (NPA).

A Service Provider has the responsibility to translate the differing requirements of the two end domains of: i) the overall network service domain, and ii) providing the required service to the end user. The goal of all parties involved is to find the best deals available in terms of quality of service, cost and penalty.

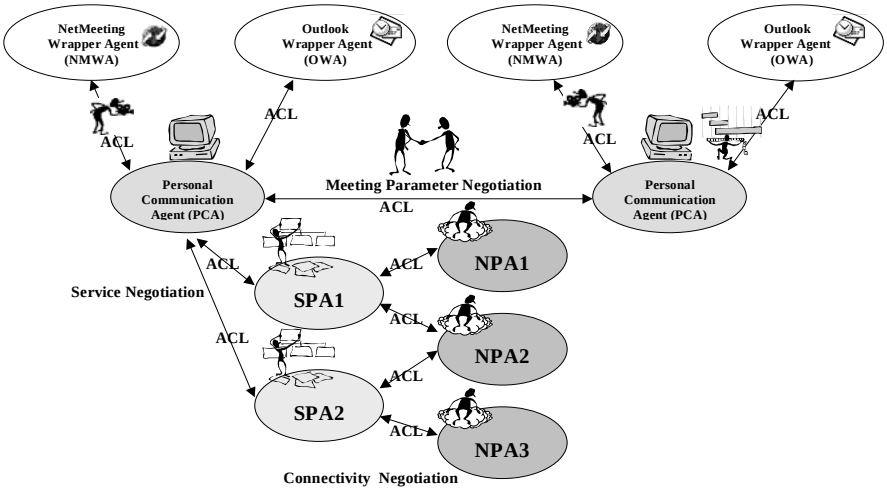

Figure 2: Inter Domain Agent Negotiation

The requirements of the service are established by the PCA with its end user. To be concrete, we consider the case in which the end user wishes to set up a videoconference meeting. The PCA contacts other users' PCAs and schedules a videoconference meeting. The initiating PCA then negotiates with the SPA to obtain the best service deal available. The SPA then re-configures the necessary parameters for the required service and negotiates with the NPAs to obtain the optimal solution.

The overall problem then is how to provision these services so that they satisfy, according to some criteria such as the price, the time or quality of the service, not only the customers but also the service providers. Achievement of this goal may require planning and solving sub problems among any number of service providers and customers. Indeed, there are often more than one service provider which both the customer and other service providers can request services or sub services from respectively. However, since each group is autonomous, services and information must be requested. For example, a service provider must somehow be persuaded to perform a service, or a service customer must be persuaded to accept a lower quality of service. Therefore the overall system can be viewed as a group of interacting and autonomous agents that provide and consume services to one another through a series of multi-lateral negotiations.

Additionally, these agents are operating in a highly dynamic environment: services need to be updated, new ones come on line, old services are removed and currently agreed services fail. Customer's requirements may also change: new services may be required, services may be required sooner or later than initially anticipated, higher quality may become more important, etc. In all of these cases, negotiation is the means of managing this complexity. New services become candidates of provisioning, those effected by the failed services can be reprovisioned, and service conditions can be dynamically configured or reconfigured. In the course of these negotiations, agents are required to make concessions to one another, to make trade-offs between the negotiation issues and to sometimes bring new issues into the negotiation in order to facilitate agreement making.

3.1 Extending the Agent Shell to construct Domain Specific Agents

All agents in the FIPA-OS distribution extend the *AgentWorldAgent* class (part of the *aw* package). The internal structure of an agent is not specified and is left to the individual agent designer. For example the following object model (figure 3) illustrates how the platform agents (AMS, DF and ACC) have been constructed by inheriting and extending the agent shell, *AgentWorldAgent*. All of the domain specific agents (PCA, SPA, NPA, AVWA, NMWA) described in this paper are constructed in a similar fashion.

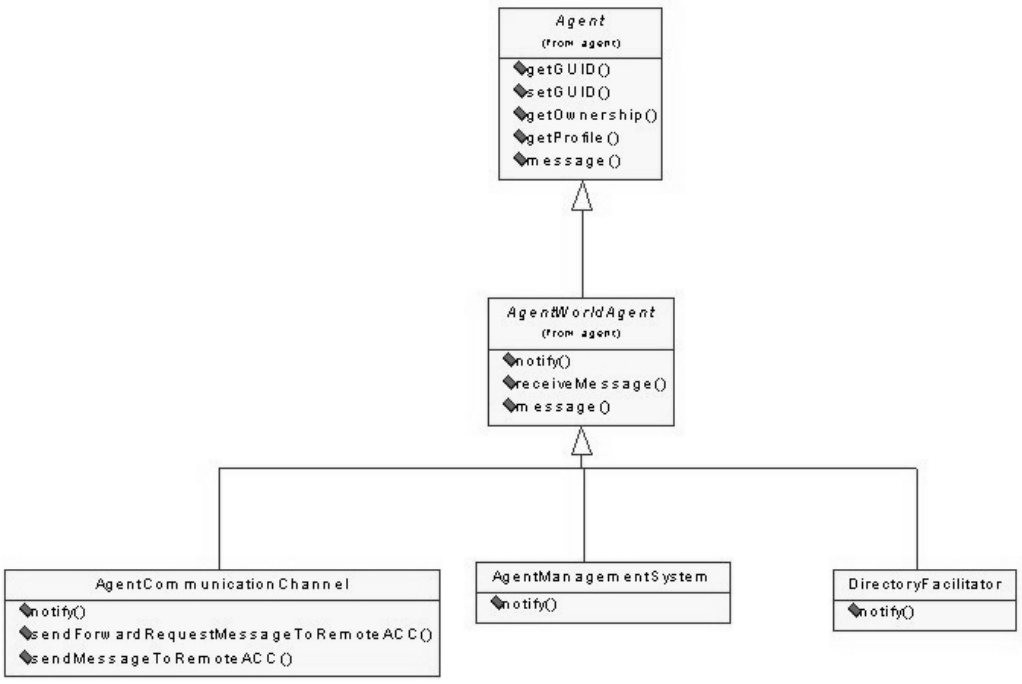

Figure 3: Object Model for FIPA-OS Platform Agents

The key roles in our application are those of the involved customers (or end-user) (sub-section 3.2), the service provider (sub-section 3.3) and the network provider (sub-section 3.4). Each of these roles are mapped to specific kinds of software agents, taking into account the following assumptions about the future network environment:

- Typically multiple competing network providers (ISPs or telecom operators), each managing a unique network domain, will allow subscribing to their connectivity services, possibly during a relatively short period of time.
- Service providers will have the task of negotiating with multiple network providers and selecting a specific one to actually deliver the service. New players could take this role, although the ISPs or operators themselves can also carry it out.

3.2 Personal Communication Agents

The Personal Communication Agent (PCA) represents the user in the system. The PCA is responsible for enacting the interests of its end user when negotiating with other PCAs for scheduling a meeting and when selecting an appropriate SPA. To do this in the best way possible, the PCAs are responsible for managing the user's profile and preferences. The PCA is also

responsible for interacting with third party software (Calendar Manager and Video Conference software in this case) at the local level (i.e. on the user's terminal). The PCA supports the following capabilities (more details of which are provided in section 4):

- FIPA agent communication protocol (FIPA ACL message handling) (FIPA 97, V2, Part 2) and interaction with FIPA agent platform components (DF, AMS and ACC) (FIPA 97, V2, Part 1).
- FIPA agent negotiation protocols (fipa-iterated-contract-net and fipa-request) and subsequent dialogue handling (FIPA 97, V2, Part 2).
- User profile management for the end users' preferences regarding scheduling meetings and videoconference (VC) configuration parameters.
- Interaction with third party software. The handling of this third party software is agent platform specific and is described in more in (FIPA 97, V2, Part3).
- Service offer negotiation capabilities for issues including: information about events, VC description, and information about the participants (including the organiser).

3.3 Service Provider Agents

The Service Provider Agent (SPA) can be considered as a VPN service broker between PCAs and NPAs. It examines the service request from the coordinating PCA and composes a service offer (including price and quality of service) based on its own service capabilities. This service offer will be the subject of the negotiation between PCA and SPA. If this negotiation has completed successfully, the SPA will subsequently start negotiations with several NPAs, which will conclude in the selection of the NPA that is most suited to provision the required network, related to the service request of the PCA. The SPA supports the following capabilities:

- FIPA agent communication protocol (FIPA ACL message handling), (FIPA 97, V2, Part 2) and interaction with FIPA agent platform components (DF, AMS and ACC), (FIPA 97, V2, Part 1).
- FIPA agent negotiation protocols (fipa-iterated-contract-net and fipa-request) and subsequent dialogue handling (FIPA 97, V2, Part 2).
- Service offer composition, based on a set of service capabilities.
- Service and network offer negotiation capabilities for issues including: event information, VC description, price, penalty, and participants.

3.4 Network Provider Agents

A Network Provider Agent (NPA) is responsible for managing its own network domain. Network offers are composed, based on network capabilities (network related quality of service parameters which can be supplied). These network offers are the subject of the negotiation between the SPA and NPA. The NPA supports the following capabilities:

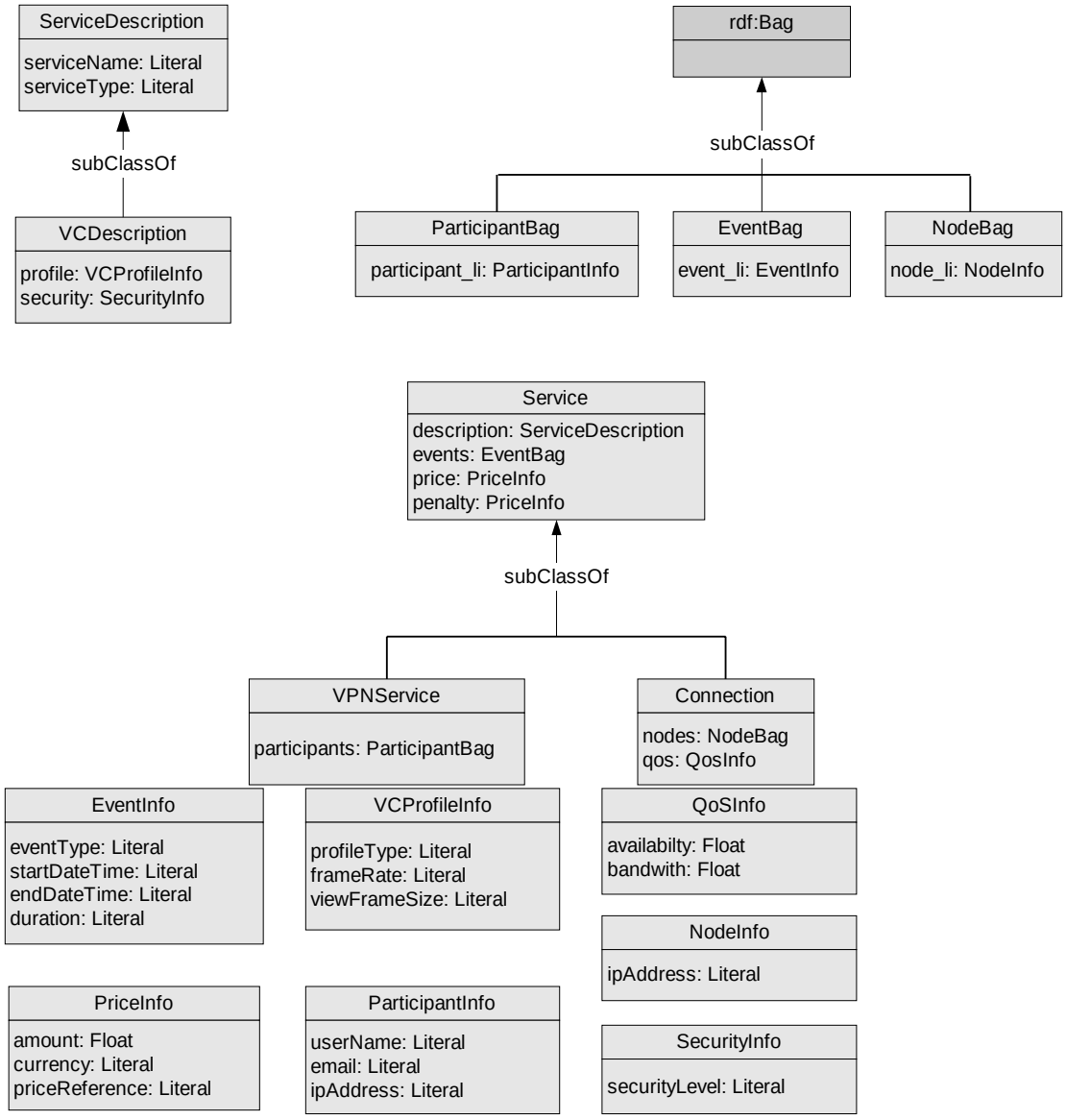

Figure 4: RDF Class Diagram for the ServiceOntology

- FIPA agent communication protocol (FIPA ACL message handling) (FIPA 97, V2, Part 2) and interaction with FIPA agent platform components (DF, AMS and ACC), (FIPA 97, V2, Part 1).
- FIPA agent negotiation protocols (fipa-iterated-contract-net and fipa-request) and subsequent dialogue handling, (FIPA 97, V2, Part 1).
- Network offer composition, based on a set of network capabilities.
- Network offer negotiation capabilities for issues including: Event information, connection description, price, and node names.

3.5 Using the FIPA ACL

Understanding an ACL message requires processing the message with regard to its temporal position within a particular interaction sequence between two or more agents. This involves understanding the type of communication called a communication act, (it is specified in the message and may be a request or statement of fact or query), understanding the structure of the content and finally understanding the semantics of the request.

As ACL communication is so rich, it is often represented as a multi-tiered layer in its own right (Finin et al, 1997). FIPA-OS supports ACL communication using four main sets of components: conversation, ACL message, content (syntax) and ontology (content semantics). Not all of these components need to be used by each agent and different combinations of different types of each these components can be supported at each layer of the FIPA-OS agent architecture. This flexibility is needed because in a heterogeneous world, different agents may encode and transport the content differently.

FIPA-OS supports both ASCII string and XML encoding of the ACL message using the appropriate decoder and parser. There are several supported encodings for the ACL message content including, FIPA SL0 and FIPA SL1 and the proposed FIPA-RDF specification (FIPA 99, Part 18) for encoding the content in XML. All agents in the scenario described in this paper use the ASCII string representation for ACL as described in (FIPA 97, V2, Part 2) and FIPA-RDF encoding for the content expressions as described in (FIPA 99, Part 18).

The VPN Service Provisioning Ontology, shown as an object model (figure 4), provides more details of the issues over which the agents negotiate. For the purposes of the negotiation described in this document, arbitrary compositions of the issues illustrated in the object model presented can be used to form the negotiable Service Level Agreements (SLAs). Moreover, the protocol for the interaction is the FIPA-Iterated-Contract-Net protocol (figure 5).

4. Implementing the Scenario

Each of the PCA, SPA and NPA agent types have been implemented using the FIPA-OS distribution. The negotiation abilities described in section 5 have been implemented as an additional component that extends the basic functionality of the Agent Shell. The functionality of each of the implemented agents is discussed below.

4.1 PCA

PCAs act on behalf of users to arrange mutually convenient meetings (in terms of time and perhaps location) for all participants with the minimum of interaction from the users. It should be stressed that although the user interaction is minimised, the final decision-making process should be left to the user. When the PCA is started, it carries out a number of tasks before it is ready to interact with other agents. Specifically, it:

- Registers itself with the AMS.
- Registers itself with the DF (This allows other agents to look up the PCA's Globally Unique Identifier (GUID) (FIPA 97, V2, Part 1) so that they can send ACL messages to it). The DF-description used by the PCA contains the agent type "**facts-pca**", service type "**pca**", service name "**pca**". The ownership is set to be the email address of the user to which the PCA represents.

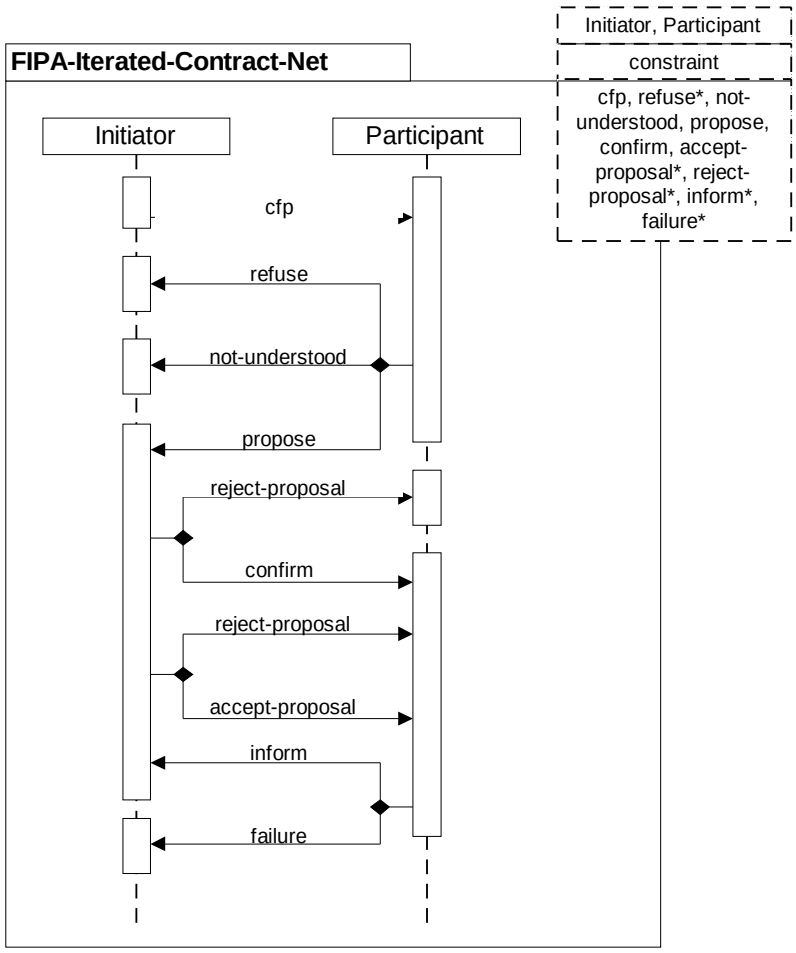

Figure 5: The FIPA Standard Iterated Contract Net Protocol

- Searches the DF for the existence of an Outlook Calendar Wrapper Agent (OWA) that belongs to the user.
- Searches the DF for the existence of a NetMeeting Wrapper Agent (NMWA) that belongs to the user.

4.1.1 Communication with OWA and NMWA

The PCA is not itself designed to integrate with legacy systems, instead this task is left to wrapper agents that can be utilised to carry out the necessary interactions. The two wrapper agents used by the PCA are the OWA and NMWA, which, respectively, provide access to the user's calendar and videoconferencing tools.

4.1.2 Negotiation with other PCA's to schedule meetings

Upon receipt of an input from the end user indicating that he wishes to arrange a meeting, the PCA initiates a negotiation (initiating PCA is referred to as the IPCA) with the invitees' "remote" PCA's (referred to as the RPCA's). This negotiation can be split into five distinct stages:

- **IPCA checks its calendar**
  The IPCA first ensures the user is available at the times they wish to attempt to book a meeting. This is accomplished by querying the user's OWA. If they are not free, then the negotiation phase halts and an error message is generated.

- **IPCA sends call for proposals to all RPCAs**
  The IPCA determines which PCAs belong to the invited users by interrogating the DF and sends a call for proposals (CFP) to each, including a range of possible times as to when to hold the meeting. The CFP also contains information about the other participants, although their specific details will not be yet known (e.g. what IP address they would be joining a videoconference from). When initiating the negotiation with the RPCAs the ACL message encoding the CFP indicates the required Interaction Protocol (FIPA 99, Part 2), which constrains the expected performatives to be used in each message exchange. In this case, it is the FIPA-Iterated-Contract-Net protocol.
- **RPCAs send proposals back to the IPCA**
  The RPCA's response is based upon: the CFP, the preferences of the user (e.g. what times of day they wish to be available for meetings) and, the times the user is available during the ranges suggested in the CFP. This process results in either a "propose" with times the user is available or a "failure"/"refuse" being sent back to the IPCA. In the case of the "propose" message, more detailed information about the participant is sent back (i.e. the IP address from which the user will join the meeting and the user's video preferences). In the case of the "refuse"/"failure" message (either due to the OWA refusing or failing to deal with the queries, or because the user is not available at all) the OWA is closed.
- **IPCA evaluates the proposals**
  Once the IPCA has received a response from each of the RPCAs, providing that at least one RPCA has made a proposal, the actual meeting scheduling process is executed. Otherwise the IPCA closes its OWA, and informs the user of the outcome.
- **IPCA sends accept/reject messages to the RPCAs**
  If a meeting cannot be arranged given the proposals made, "reject" messages are sent to all of the RPCA's to indicate that their proposals have been rejected. If a meeting could be arranged, "reject" messages are just sent to the RPCA's whose users cannot attend and "accept" messages are sent to the RPCA's whose users can (along with the precise details of the meeting, and the other participants who will be attending.

4.1.3 Negotiation with SPAs to arrange VPN provisioning

Once a meeting time has been arranged and at least one RPCA has responded favourably to the "accept" message, the PCA attempts to locate a number of SPAs by sending a query to the DF. If none can be located, then the PCA will send a response through the PCA GUI, indicating the arranged meeting time and the participants who can attend, but will include a warning message indicating that a VPN has not been provisioned for the time of the meeting.

Should at least one SPA be found, the IPCA will begin negotiating with the SPAs over the quality of service (QoS) required for the meeting (i.e. the PCA will aim to get the closest QoS to that defined by each user's VCProfile, which were exchanged during the meeting scheduling). This negotiation follows the FIPA-Iterated-Contract-Net protocol, and the PCA has a mechanism to generate counter-proposals based upon the proposals made by the SPAs and the QoS target it is trying to reach. Once a compromise has been reached between the SPA and PCA and an agreement has been made, the PCA then informs the user, through the GUI, of the details of the arranged meeting.

4.1.4 Starting the videoconference once the VPN has been commissioned

Upon receiving a request from an SPA indicating that the VPN is being commissioned, the PCA takes the following steps:

- **IPCA configures local A/V conferencing application to host the meeting**
  The user's NMWA is initialised, and then sent the "start" command to configure the videoconferencing application(s) (i.e. NetMeeting/CamWiz) to host the meeting. If this fails an error message is displayed for the user.
- **IPCA informs RPCA's that the conference is ready**
  The IPCA will perform a DF look-up of the necessary RPCAs, then send them a "request" message with the details of the meeting. Upon receipt of this message, an RPCA will initialise the participating user's AVWA, and send it a "call" command in order that the videoconferencing application(s) should call into the conference that the IPCA's associated AVWA has initiated. Once this has been completed successfully, the RPCA replies with an inform/done message to indicate success, otherwise it will respond with a refuse/failure message based upon the reason why the user cannot join the conference.

4.2 SPA

SPAs negotiate with an IPCA regarding the state and conditions (e.g. type of service, starting time of the service, price, penalty) of the service that the SPA is to provide. It then negotiates with collaborating NPAs to find the best solution for the provisioning of the service to the customer. An SPA has an interest in maximising its profit.

4.2.1 Register with AMS and DF

The SPA must register with an AMS and a DF before it can function. The current version of the SPA registers itself in the following manner: agent type **"facts-spa"**, ownership **"www.agentworld.co.uk"**, service type **"spa"** and service name **"spa**".

4.2.2 Negotiates with PCA

When a meeting is agreed among the PCAs, the IPCA sends the SPAs a service request. The IPCA starts the negotiation over service characteristics (including price, frame rate, view frame size and penalty). The negotiation may consist of a number of iterations following the FIPA-Iterated-Negotiation protocol until the negotiation either succeeds or fails. The negotiation fails when the maximum time for negotiation of the IPCA or SPA is reached. The negotiation succeeds when the proposed service characteristics are agreed between the IPCA and an SPA.

4.2.3 Queries the DF for information on available NPAs

When the service characteristics are agreed, the SPA send a request to the DF for a search on the information of available NPAs.

4.2.4 Negotiates with NPA(s)

When the DF replies with a list of NPAs, the SPA calculates the necessary bandwidth for the agreed service, and re-configures the minimum and maximum values for price and penalty for negotiation to ensure a minimum profit. The SPA then sends each of the NPAs a "call for proposals" (CFP) message, and starts SPA-NPA negotiations. An example CFP message with a subset of the issues presented in figure 4 encoded in XML is given below:

```
(cfp
  :sender spa3-spa@iiop://195.8.93.19:50/acc
  :receiver npa5-npa@iiop://195.8.93.19:50/acc
  :content
    (<?xml version="1.0"?>
    <rdf:RDF   xml:lang="en"
               xmlns:so="http://193.121.106.20:8001/schemas/ServiceOntology#"
               xmlns:st="http://193.121.106.20:8001/schemas/ServiceTransaction#"
               xmlns:fipa="http://193.121.106.20:8001/schemas/FipaSchema#"
               xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <st:proposeConnection rdf:ID="initialCFP">
         <fipa:actor>SPA</fipa:actor>
         <fipa:argument rdf:resource="VPNService"/>
    </st:proposeConnection>
    <so:Connection rdf:about="VPNService">
         <so:description rdf:resource="vcdescription"/>
         <so:events rdf:resource="events"/>
         <so:price rdf:resource="price1"/>
         <so:qos rdf:resource="qos1"/>
    </so:Connection>
    <so:VCDescription rdf:about="vcdescription">
       <so:serviceType>VideoConference</so:serviceType>
       <so:serviceName>Netmeeting</so:serviceName>
       <so:profile rdf:resource="vcprofile"/>
    </so:VCDescription>
    <so:VCProfileInfo rdf:about="vcprofile">
       <so:profileType>ViedoConference</so:profileType>
       <so:frameRate>Med</so:frameRate>
       <so:viewFrameSize>Small</so:viewFrameSize>
    </so:VCProfileInfo>
    <so:EventBag rdf:about="events">
         <so:event_li rdf:resource="event1"/>
    </so:EventBag>
    <so:EventInfo rdf:about="event1">
         <so:startDateTime>19990708T111100000Z</so:startDateTime>
         <so:duration>+00000000T010000000</so:duration>
    </so:EventInfo>
    <so:PriceInfo rdf:about="price1">
         <so:amount>9.377499</so:amount>
         <so:currency rdf:resource="http://bt00sz:8001/schemas/UnitSchema#Euro"/>
         <so:priceReference rdf:resource="http://bt00sz:8001/schemas/UnitSchema#Second"/>
    </so:PriceInfo>
    <so:QoSInfo rdf:about="qos1">
        <so:availability>85.0</so:availability>
        <so:bandwidth>7.4600005</so:bandwidth>
    </so:QoSInfo>
    </rdf:RDF>)
  :reply-with spa3-spa0
  :in-reply-to spa3-spa0
  :language FIPA-RDF
  :ontology vpn-service
  :protocol fipa-iterated-contract-net
  :conversation-id 931452074937)
```

If all of the SPA-NPA negotiations fail, the SPA will inform the IPCA that the agreed service can not be provided. However, if one of the SPA-NPA negotiations succeeds, the SPA will confirm with the PCA about the agreed service and it will then set its internal timer to alarm at the agreed start time.

4.2.5 Informs PCA of starting time

When the start time of the meeting is agreed, the SPA will request that the NPA provisions the agreed VPN and that the IPCA starts the agreed AV conference.

4.3 NPA

NPAs are responsible for the provisioning of the network connectivity upon requests from the SPA. NPAs negotiate with the SPA about terms and conditions of network connectivity. A Network Provider has an interest in maximising its profit.

4.3.1 NPA Registers with AMS and DF

NPAs must register with an AMS and a DF before they can function. The current version of the NPA registers itself with a agent type **"facts-npa"**, ownership **"www.agentworld.co.uk"**, service type **"npa"** and service name **"npa**".

4.3.2 NPA Negotiates with SPA

When service provisioning is agreed between the SPA and IPCA, the SPA sends a selection of NPAs a connection request (cfp). SPA starts the negotiation over connectivity characteristics (include price, bandwidth, availability and penalty). The negotiation may consist of a number of iterations following the FIPA-Iterated-Negotiation protocol until the negotiation succeeds or fails (when the maximum time for negotiation of the SPA or NPA has been reached). The negotiation succeeds when the proposed connection characteristics are agreed by the SPA.

5. Agent Negotiation

The previous sections have explained the solution technology that addresses the ACL and the engineering problem for a MAS. This section focuses on the coordination negotiation element of the service provisioning aspect of our system. In particular, we focus on the reasoning model that the *VPN* agents employ in order to reach agreements for providing and consuming services from one another. The model is based on a formal definition, the details and rationale for the design choices for which are contained in (Faratin *et al*, 1998, Faratin *et al*, 1999, Faratin *et al*, 2000, Jennings *et al*, 2000, Sierra *et al*, 1999).

The decisions faced by our agents are a combination of:

- *offer generation decisions* (what initial offer should be generated? what counter offer should be given in situations where the opponent's offer is unacceptable?), and
- *evaluatory decisions* (when should negotiation be abandoned? and when should an agreement be deemed to have been reached?).

The solution to these decision problems is captured in the agent architecture (figure 6). The evaluatory decision (Eval components in figure 6) is expanded on in (Faratin *et al* 1999). The offer generation components (or what is referred to as the mechanisms) of the architecture (Responsive, Tradeoff and Issue man. components in figure 6) are distinguished by the following properties:

- the computational and informational cost the mechanism incurs on the agent
- the social benefit of the mechanism for the agents

The first property is a feature that distinguishes our work from much of the work on game theory models (e.g. (Rosenchein and Zlotkin, 1994)). The design of the *VPN* negotiating agent is highly constrained by the fact that in real world applications agents are seldom computationally and informationally unbounded, an assumption which underpins many of the classic game theory models. The provisioning of a *VPN* service is a real time process. Services are required within tight scheduling windows and a negotiation mechanism must respect the time limits of the agent. Furthermore, negotiation is only a single element of the agent's deliberations and so it must not consume disproportionate amounts of computational resource. In addition to being computationally bounded, agents are also information bound. For example, an *IPCA* agent will

not know the preferences, resources or execution time lines of a *SPA* agent. Thus a negotiation mechanism must find solutions when information is private.

The second feature relates to the concern for the design of a mechanism that achieves some measure of social (or global) coherency from local and distributed processing. Since *VPN* agents are economical agents then one such measure of social coherency is the sum of the value for each of the agents for an outcome (c.f. pareto optimal deals (Gorfman *et al*, 1993)). Nash is another, (Gorfman *et al*, 1993). Using this feature, we can distinguish between mechanisms that are concerned with *individual utility* of the outcomes without concern for the social welfare, and ones that produce outcomes that are both individually *and* jointly preferred by the agents. For example, if a deal is required very soon then negotiation between the *IPCA* and *SPA* agents is driven by concern for a deal that is perhaps not socially optimal but one that is agreeable by both agents. On the other hand, if there is time to negotiate, then the same negotiation may involve both agents searching for deals that are not only individually rational but may also be beneficial to the other agent.

Given these properties we have designed and implemented three negotiation mechanisms (responsive, trade-off and issue manipulation mechanisms) which differentially implement these properties. Figure 6 describes the functional model of the agent's reasoning during negotiation. Given the negotiation deadline (*Tmax*), the opponent's last offer (*Y*) and the agent's last offer (*X*) (the three inputs into the negotiation architecture in figure 6) all three mechanisms simultaneously compute a new offer (*X')* (boxes Responsive, Trade-off and Issue man, in figure 6). The mechanism's evaluatory component then makes the decision to either *accept* (*A*) or *reject* (*R*) the opponent's last offer *Y*, or *counteroffer* (X'), the new contract, to the opponent. The final choice of which mechanism's suggestion to offer is handled by the *meta-strategy* module. The processes involved in each mechanism are described next.

5.1 Responsive Mechanisms

Responsive mechanisms model reactive negotiation behaviour to a number of environmental factors. The underlying rationale and motivation for the design of these mechanisms has been the need to model concessionary behaviours that are initiated by progressively more important environmental factors during the course of negotiation process. For example, if *IPCA* has committed many resources during its negotiation with *SPA* and the time of the required video service with other *RPCAs* is soon, then the *PCA* may prefer simple and less costly decision mechanisms that can result in concessions. Concession may result in an agreement, and therefore not only free *IPCA's* resources, which can be used for other activities, but also achieve the goal of establishing a meeting with the *RPCA*s.

Responsive mechanisms generate offers by linearly combining simple decay functions, called *tactics* (Faratin *et al*, 1999). Tactics generate values for issues using only a single environmental criterion. We have designed three families of tactics:

- **Time-dependent tactics**: model increasing levels of concession as the deadline for the negotiation approaches;
- **Resource-dependent tactics**: model increasing levels of concession with diminishing levels of resources;
- **Behaviour-dependent tactics**: model concessions based on the behaviour of the other negotiating party.

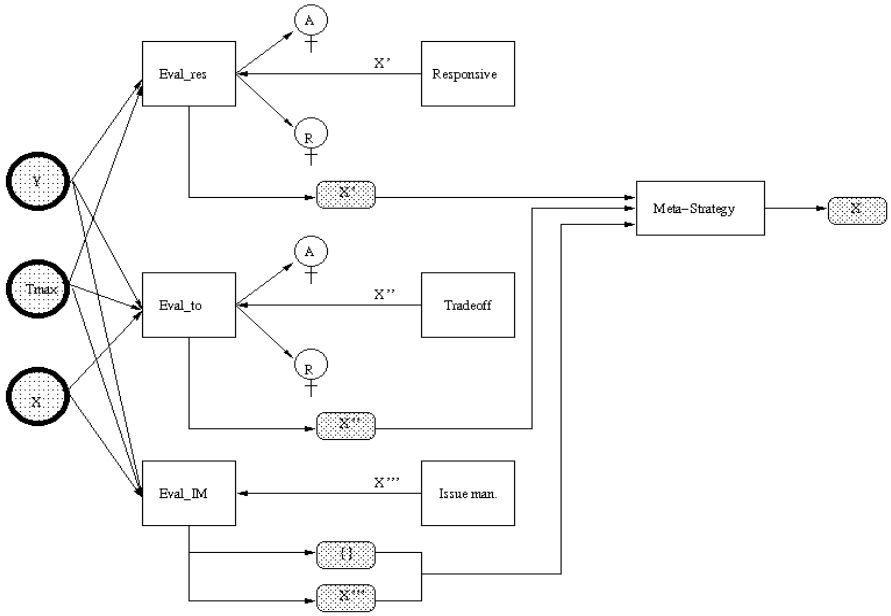

Figure 6: The Negotiation Architecture

To determine the best course of action, an agent may need to consider and assess more than just one environmental condition. Since each tactic generates a value for an issue using only a single criterion, the concept of *strategy* is introduced to model the modification, over time, of tactic weights as the criteria change their relative importance in response to environmental changes.

5.2 Trade-off Mechanisms

A concession mechanism generates contracts that are individually rational. That is, given the current state of the environment, the mechanism computes what is the best offer an agent can propose that maximises *its* value. However, in some cases there is also a need for agents to act in a more socially responsible manner. Thus, for example, negotiation between *NPAs* to jointly provide an IS service to a *SPA* agent is naturally a cooperative activity when the *NPAs* represent the same network operator. In such cases, the agents are concerned both with the outcome of the negotiation for themselves and for their negotiation opponent. In short, they care about equity and social welfare (Gorfman *et al*, 1993), as well as their individual utility. A concession mechanism in such contexts is inefficient, in terms of possible joint value gains (Rosenchein and Zlotkin, 1994). This requirement led us to design models that can uncover win-win negotiation solutions (Raiffa, 1992), again in the presence of limited knowledge and computational boundedness. Win-Win negotiation refers to a bargaining situation where both parties search for solutions that "squeeze out'' more gains (either mutually or individually) than the currently agreed deal.

The particular mechanism for win-win negotiation that we developed in this context is that of agents making *trade-offs* (Faratin *et al*., 2000). Intuitively, a trade-off is where one party lowers its scores on some negotiation issues and simultaneously demands more on others. Thus, an *NPA* may accept a service of lower quality if it is cheaper, or a shorter deadline if it receives a higher price. Such movements are intended to generate an offer that, although of the same value to the proposer, may benefit the negotiation opponent and hence increase the overall gains between the two agents.

An agent will decide to make a trade-off action when it does not wish to decrease its aspirational level (denoted θ) for a given service-oriented negotiation. Thus, the agent first needs to generate some/all of the potential contracts for which it receives the score of θ. Technically, it needs to generate contracts that lie on the iso-value (or indifference) curve for θ (Raiffa, 1992). Because all these potential contracts have the same value for the agent, it is indifferent amongst them. Given this fact, the aim of the trade-off mechanism is to find the contract that is most preferable (and hence acceptable) to the negotiation opponent since this maximises the joint gain. More formally, an iso-curve is defined as follows: given an aspirational scoring value θ, the iso-curve set at level θ for agent *a* is defined as:

$$Iso_a(\theta) = \{ x \mid V_a(x) = \theta \}$$

where $V_a(x)$ is the value of contract *x* for agent *a*. From this set, the agent needs to select the contract that maximises the joint gain. However, since an agent does not know its opponent's utility function some form of approximation is needed. The heuristic we employ is to select the contract that is most "similar" to the opponent's last proposal (since this might be more acceptable to the opponent). To compute similarity we use the concept of fuzzy similarity, (Zadeh, 1971). Fuzzy similarity is an approximation heuristic, which supports reasoning of the kind "*if p is true, then q is close to being true*". This notion was chosen because it allows an agent to approximately model the closeness of two contracts in decision making.

A trade-off is defined as follows. Given an offer, *x*, from agent *a* to *b*, and a subsequent counter offer, *y*, from agent *b* to *a*, with $\theta = Va(x)$, a trade off for sent *a* with respect to *y* is defined as:

$$trade\text{-}off_a(x,y) = argmax\ z \in iso_a(\theta)\ \{ Sim(z,y) \}$$

where the similarity, *Sim*, between two contracts is defined as a weighted combination of the similarity of the issues. The similarity between two contracts *x* and *y* over the set of issues *J* is defined as:

$$Sim(x,y) = \sum_{j \in J} w^a_j\ Sim_j(x_j,y_j)$$

with $\sum\{j \in J\}\ w^a_j = 1$ and Sim_j being the similarity function for issue *j*.

5.3 Issue Manipulation Mechanisms

Our other deliberation mechanism is the issue set manipulation (Faratin *et al*., 1999). Negotiation processes are directed and centred around the resolution of conflicts over a set of issues *J*. This set may consist of one or more issues (distributed and integrative bargaining respectively (Raiffa, 1992)). For simplification we assume the ontology of the set of possible negotiation issues (see figure 4) *J*, is shared knowledge amongst the agents. It is further assumed that agents begin negotiation with a pre-specified set of "core" issues, $J(core) \subseteq J$, and possibly other mutually agreed non-core set members, $J(\neg core\} \subseteq J$. Alterations to *J(core)* are not permitted since some features such as the *price* of services are mandatory. However, elements of $J(\neg core\}$ can be altered dynamically. Agents can add or remove issues into $J(\neg core\}$ as they search for new possible and up to now unconsidered solutions.

In the VPN scenario, agents negotiate over both core and non-core issues. Moreover, it is sometimes important to be able to change the set of issues in order to make an agreement more likely or relevant. For example, a *SPA* may begin *QoS* negotiation with a *NPA* specifying only *bandwidth*. However, subsequently *NPA* may decide to include into the *QoS* negotiation a *packetloss* issue with a high value if *SPA* has demanded a high capacity *bandwidth*. Alternatively,

SPA may decide to remove the *bandwidth* issue from the *QoS* negotiation with *NPA* if *IPCA* has changed its demand from a high quality video service to a standard audio service. Agents deliberate over how to combine these *add* and *remove* operators in a manner that maximises some measure --- such as the contract score (Faratin *et al*, 1999). However, a search of the tree of possible operators to find the optimum set of issues may be computationally expensive. To overcome this problem we are in the process of implementing anytime algorithms that use the negotiation time limits to compute a, possibly sub-optimal, solution.

5.4 Meta Strategies

The above-mentioned mechanisms formally capture the decision problems of *VPN* agents in the course of service negotiation. However, since there are three choices of mechanism an agent is now faced with the decision problem of which of the three to choose from. The function of the meta-strategy reasoner is to resolve this decision problem by selecting one of the offers of the responsive, trade-off or issue manipulation mechanisms. The meta-strategy of which mechanism to select can be based on any number of decision factors such as the time deadline to reach a solution in negotiation, the concern for global optimality of the negotiated solution, the behaviour of the opponent in negotiation, negotiation is in stalemate, etc. For example, if the time to reach a solution is adequate, *VPN* agents may adopt a meta-strategy that utilises the trade-off mechanism to search for a negotiation deal that increases the global benefit. On the other hand, if the time to reach a solution is very short, then a responsive mechanism is more likely to find agreements that, although are of lower joint utility, are nonetheless better than no deal. Alternatively, if negotiation is perceived to be in deadlock then modification of the set of issues in negotiation may break deadlocks. The choice rule of which mechanism to select is currently being empirically investigated in a number of different negotiation environments.

6. Conclusions

This paper has described the design and implementation of a multi-agent system for provisioning virtual private networks. The implementation is based upon the FIPA standard and was realised through the FIPA-OS platform. A key component of this application was the automated negotiation that took place between the various stakeholder agents. To this end, we described the negotiation algorithms we have developed and implemented for this application. Our algorithms enable agents to engage in flexible negotiations and incorporate structures for concession making in the face of varying environmental factors, issue trade-off mechanisms and issue manipulation mechanisms.

REFERENCES

A. Bond and L. Gasser (1988) *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers, Inc., San Mateo, California.

K.P. Corfman and S. Gupta (1993) *Mathematical models of group choice and negotiations.* Handbook in Operations Research and Management Sciences, 5, 83-142.

P. Faratin, C. Sierra, and N. R. Jennings (1998) *Negotiation Decision Functions for Autonomous Agents*. Int. Journal of Robotics and Autonomous Systems 24 (3-4) 159-182

P. Faratin, C. Sierra, N. R. Jennings and P. Buckle (1999) *Designing Responsive and Deliberative Automated Negotiators*. Proc. AAAI Workshop on Negotiation: Settling Conflicts and Identifying Opportunities, Orlando, FL, 12-18.

P. Faratin, C. Sierra, and N. R. Jennings (2000) *Using Similarity-Criteria to Make Negotiation Trade-Offs.* Proc. Int Conf. On Multi-Agent Systems (ICMAS-2000), Boston, USA.

FACTS URL: http://www.labs.bt.com/profsoc/facts/

T. Finin, Y. Labrou and J. Mayfield (1997) *KQML as an agent communication language*. In Software Agents, Bradshaw JM (ed.), MIT Press, 291-316.

FIPA-OS URL: http://www.nortelnetworks.com/fipa-os

FIPA97 V2, Part 2 *Agent Communications*. October 1998 URL: http://www.fipa.org/spec/fipa97.html

FIPA97 V2, Part 1 *Agent Management.* October 1998 URL: http://www.fipa.org/spec/fipa97.html

FIPA97 Part 3 *Agent/Software Integration*. October 1997 URL: http://www.fipa.org/spec/fipa97.html

FIPA98, Part 13 *Developer's Guide* October 1998 URL: http://www.fipa.org/spec/fipa98.html

FIPA99, Part 18 *Content Language library* January 2000 URL: http://www.fipa.org/spec/fipa99.html

FIPA99, Part 2 *Agent Communication* January 2000 URL: http://www.fipa.org/spec/fipa99.html

N. R. Jennings, P. Faratin, T. J. Norman, P. O'Brien and B. Odgers (2000) *Autonomous Agents for Business Process Management.* Int. Journal of Applied Artificial Intelligence 14 (2).

H. Raiffa (1982) *The Art and Science of Negotiation*. Harvard University Press, Cambridge, USA.

J.S. Rosenchein and G. Zlotkin (1994) *Rules of Encounter*. The MIT Press, Cambridge, USA.

C. Sierra, P. Faratin and N. R. Jennings (1999) *Deliberative Automated Negotiators Using Fuzzy Similarity*. Proc EUSFLAT-ESTYLF Joint Conf. on Fuzzy Logic, Palma de Mallorca, Spain, 155-158.

G. Weiss (1999) *Multiagent Systems*, The MIT Press, Cambridge, Massachusetts.

L.A. Zadeh (1971) *Similarity Relations and Fuzzy Orderings.* Information Sciences, 3,177-200.